IMDb Movies Review Prediction

Yuki Kitamura (UID: 305535199), Helena Xu (UID: 105513264),
Cherry Li (UID: 505943441), Bonnie Gu (UID: 005588759),
and Joshua Susanto (UID: 405568250)

Department of Statistics and Data Science

Abstract

IMDb is an online database for movies, TV shows, and celebrity content. In this project, we focus on the movie reviews given by IMDb users, which have two components: written reviews and sentiments (positive or negative). We will analyze the relationship between the written reviews and the sentiments to classify a movie sentiment as either positive or negative from a written review by using the following models: logistic regression, k-nearest neighbors (KNN), linear discriminant analysis (LDA), and quadratic discriminant analysis (QDA).

1 Introduction

Looking for a new movie to watch? Want some recommendations based on movies you have enjoyed? IMDb is an online database with ratings, reviews, and information on where to watch the best movies, TV shows, and other streaming content online. With IMDb being the world's most popular and authoritative source for movies, many consumers have relied on this platform to choose a movie best suited for them to watch. Each IMDb review contains a numerical rating from a 10-star rating system and a description of the user's experience of the movie.

In our project, we are tasked with analyzing a IMDb movie review dataset with 50,000 reviews. Though instead of a numerical rating, each review has a corresponding sentiment (positive or negative) in regards to the user's overall experience of the movie. We aim to explore the relationship between reviews and sentiments through textual analysis and ultimately create a model that will classify a movie sentiment as either positive or negative from a written review.

2 Data Preprocessing

Our raw data was given in the form of a csv file, where each observation contains the written review and its corresponding sentiment. We first performed some exploratory data analysis to gain a better understanding of our data. Afterward, we transformed each review into a numerical vector using a dictionary. We used two dictionaries, one created by a Python package and a sentiment dictionary provided, to create two different numerical datasets for comparing model performances between two dictionaries.

2.1 Descriptive Statistics

We constructed some descriptive statistics to gain a better understanding of our data.

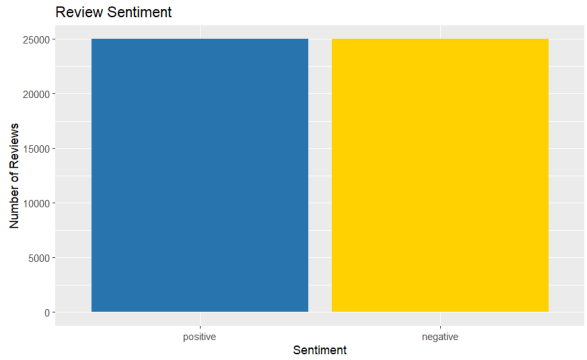

Figure 1: Distribution of Sentiments

We see that there is an equal amount of positive and negative sentiment reviews in the dataset.

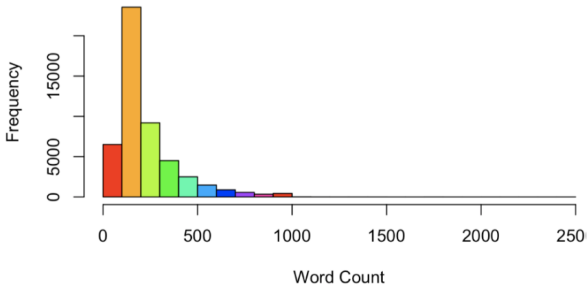

Figure 2: Distribution of Word Count

We can see that the distribution of word count is highly right-skewed. The majority of the reviews are under 200 words as the peak of the distribution is at 100 to 200 words.

Table 1: Summary Statistics of Word Count

Max	Min	Mean	Median
2450	4	227	170

There is extremely high variability in the word counts of reviews, since the difference between maximum and minimum is 2446 words. The median is lower than the mean, which reflects the right-skewed distribution of word counts.

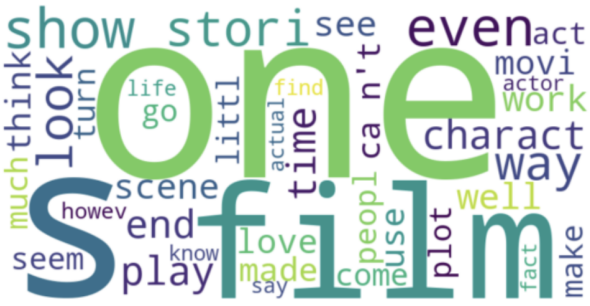

Figure 3: Frequency of Words in Reviews

We can see that there are no positive or negative words in the word cloud. All the terms are related to movies, common verbs, or simply stop words; therefore, removing these not-useful common words can make the numericalization of the review easier.

2.2 Numericalization of Reviews

With the raw dataset, we have a problem that a written review cannot be directly run through any classification model to predict the sentiment of the review. To use a classification model, we must find a way to transfer a written review into a numeric vector.

One method to numerically vectorize a written text is to use the term frequency and inverse document frequency (TF-IDF) of a predefined dictionary. A text will convert into a vector, a list of numbers greater than or equal to 0, where each number is a measure of the importance of each dictionary word used in the text.

We first created our dictionary by analyzing each term used in all reviews from the dataset. We used a package from Python called "TfidfVectorizer", which converts a collection of text documents to a matrix of TF-IDF features. When the package is run to a collection of text documents, it automatically removes stop words, words that are commonly used but have no meaning, such as personal pronouns, determiners, conjunctions, and prepositions. We used the package on all 50,000 reviews by considering the top 2,000 features. It out-

putted the TF-IDF matrix of the top 2,000 frequently used words, where each row is a TF-IDF of 2,000 terms of each review. This TF-IDF matrix is used directly as an input for classification models.

Our dictionary, the top 2,000 words, contains positive and negative sentiments in addition to some non-sentimental words, such as 90s, some verbs and nouns. We decided to use this as our dictionary and let the model decide if each word is a positive or negative sentiment rather than creating a dictionary with only positive and negative sentiments.

Besides using our original dictionary for modeling, we decided to run classification with the sentiment dictionary provided to compare the performances. Similar to what we did with our original dictionary, we created TF-IDF matrix of all 6,786 terms in the sentiment dictionary. The TF-IDF matrix is used as a second dataset for classification models to compare performances between our dictionary and the sentiment dictionary.

2.3 Feature Engineering: Principal Component Analysis (PCA)

Principal component analysis (PCA) is a variable dimension reduction approach, which finds new variables with less than the original number of variables that are independent of one another and contain the most variance.

We decided to perform PCA on our dataset since the number of columns is very large from the dictionary, each column can be highly correlated to each other, and it is computationally expensive to run the raw dataset into different models.

To determine the best number of predictors for the TF-IDF matrix of our dictionary, we calculated the explained variance of all 2,000 components in order and then plotted their cumulative explained variance by each component.

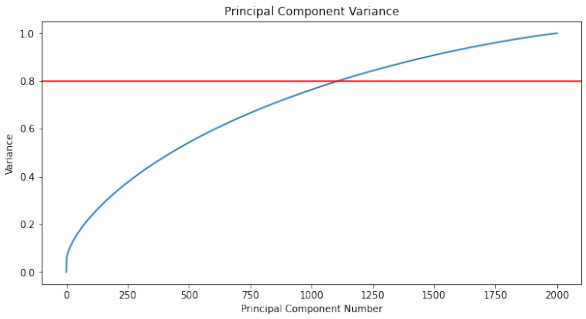

Figure 4: Principal Components by Proportion of Variance

We can see in Figure 4 that around 90% of the variance can be explained by the first 1,500 components, and 80% of the variance can be explained by the first 1,100 components. Since running 1,500 components to modeling is still computationally expensive, we decided to perform PCA with 1,100 components, which explains a good percentage of the variance with a good number of component reductions.

Similarly, we determined the best number of predictors for the TF-IDF matrix of the sentiment dictionary. We calculated the explained variance of all 6,786 components in order and then plotted their cumulative explained variance by each component.

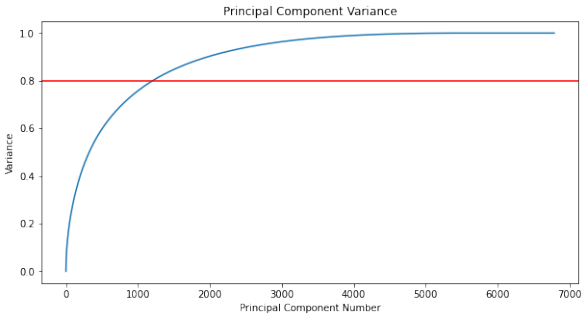

Figure 5: Principal Components by Proportion of Variance (Sentiment Lexicon Features)

Figure 5 shows that 80% of the variance can be explained by the first 1,100 components, which is the same as the PCA for the original dictionary. For the sentiment dictionary, PCA of 1,100 components is chosen to run modeling.

3 Experiment

To determine the best prediction model, we compared three different models: logistic regression, linear and quadratic discriminant analysis, and k-nearest neighbors. In addition, we fit logistic regression on four different datasets: TF-IDF matrix of the top 2,000 terms from our dictionary, TF-IDF matrix of the top 2,000 words from the sentiment dictionary, TF-IDF matrix of the top 5,000 words from the sentiment dictionary, and TF-IDF matrix of the top 5,000 terms from our dictionary to evaluate the accuracy performance among different number of predictors and dictionary. Within these comparisons, we also compared the performance of models fitted with and without feature engineering with PCA.

3.1 Logistic Regression with 2,000 Terms with Original Dictionary

Logistic regression, which is an extension of linear regression that is used for predicting continuous variables, is a method used to estimate the probability that a binary instance belongs to a given class.

For our model using 2,000 terms with our original dictionary, we first converted the labels to numeric values, with positive having a value of 1 and negative having a value of 0. We then split the full dataset into the training and testing datasets, and proceeded to initialize and fit the logistic regression model on the training data. We implemented the model to make predictions on the testing data, which yielded an accuracy of 0.8736.

Table 2 below shows the results we received from the test data after we trained the logistic regression model.

Table 2: Logistic Regression (2,000 Terms with Original Dictionary) Test Results

	Precision	**Recall**	**F1-Score**	**Support**
Positive	0.88	0.86	0.87	12483
Negative	0.86	0.89	0.88	12517
Accuracy	–	–	**0.87360**	25000
Macro Avg	0.87	0.87	0.87	25000
Weighted Avg	0.87	0.87	0.87	25000

3.2 Logistic Regression with All Terms From Sentiment Dictionary

We also performed logistic regression using all terms from the sentiment dictionary.

Table 3: Logistic Regression (All Terms From Sentiment Dictionary with PCA) Test Results

	Precision	**Recall**	**F1-Score**	**Support**
Positive	0.87	0.84	0.85	12483
Negative	0.84	0.87	0.86	12517
Accuracy	–	–	**0.85440**	25000
Macro Avg	0.85	0.85	0.85	25000
Weighted Avg	0.85	0.85	0.85	25000

In this case, the precision, recall, and overall accuracy are slightly lower than that of the logistic regression model that uses 2,000 terms with the original dictionary.

3.3 Logistic Regression with 2,000 Terms with Original Dictionary (Feature Engineered with PCA)

Next, we used 2,000 terms with the original dictionary on logistic regression, after using PCA for feature engineering.

Table 4: Logistic Regression (2,000 Terms with Original Dictionary and PCA) Test Results

	Precision	**Recall**	**F1-Score**	**Support**
Positive	0.88	0.86	0.87	12483
Negative	0.86	0.89	0.88	12517
Accuracy	–	–	**0.87324**	25000
Macro Avg	0.87	0.87	0.87	25000
Weighted Avg	0.87	0.87	0.87	25000

Here, the accuracy of this particular logistic regression model is fairly comparable to the first logistic regression model we fitted, with the accuracy score being about 0.003 lower.

3.4 Logistic Regression with All Terms From Sentiment Dictionary (Feature Engineered with PCA)

Our final logistic regression model involved all terms from the sentiment dictionary. PCA was also implemented for feature engineering.

Table 5: Logistic Regression (All Terms From Sentiment Dictionary with PCA) Test Results

	Precision	Recall	F1-Score	Support
Positive	0.87	0.84	0.85	12483
Negative	0.84	0.87	0.86	12517
Accuracy	–	–	**0.85388**	25000
Macro Avg	0.85	0.85	0.85	25000
Weighted Avg	0.85	0.85	0.85	25000

Based on the accuracy score of 0.85388, we observe that the performance of logistic regression models is similar for both that utilize terms from the sentiment dictionary.

3.5 Linear and Quadratic Discriminant Analysis with Original Dictionary (Feature Engineered with PCA)

Both logistic regression and KNN are discriminative models, which study the conditional probability distribution P(Y—X) to model the decision boundary between the classes. Generative models, on the other hand, study the joint probability distribution distribution P(X, Y) and can generate new data instances. Two of such models that we will now explore are Linear Discriminant Analysis (LDA) and Quadratic Discriminant Analysis (QDA).

For discriminant analysis models, including LDA and QDA, we have to first make the assumption that the conditional distribution P(X=x — Y = k) is a multivariate normal distribution. However, the difference between LDA and QDA is that unlike QDA, LDA further assumes that all classes of the response variable have a common covariance matrix.

LDA and QDA classifiers are attractive because they have closed-form solutions that can be easily computed and have no hyperparameter to tune. We have fitted both LDA and QDA models to our training dataset and implemented them on our testing dataset to receive the following results:

Table 6: LDA (with Original Dictionary and PCA) Test Results

	Precision	Recall	F1-Score	Support
Positive	0.88	0.85	0.87	12483
Negative	0.85	0.89	0.87	12517
Accuracy	–	–	**0.86864**	25000
Macro Avg	0.87	0.87	0.87	25000
Weighted Avg	0.87	0.87	0.87	25000

Computation Time: 0:00:13.966696

Table 7: QDA (with Original Dictionary and PCA) Test Results

	Precision	Recall	F1-Score	Support
Positive	0.83	0.81	0.82	12483
Negative	0.82	0.83	0.82	12517
Accuracy	–	–	**0.82172**	25000
Macro Avg	0.82	0.82	0.82	25000
Weighted Avg	0.82	0.82	0.82	25000

Computation Time: 0:00:16.012062

Through these results, we can see that LDA outperformed QDA in all evaluation metrics and had an accuracy of 86.86% in comparison to the 82.17% accuracy for QDA. A reasonable explanation for this outcome is that the additional flexibility of QDA was unnecessary in this case and only resulted in a higher variance.

Since both logistic regression and LDA look for a linear decision boundary (i.e. hyperplane) to separate the classes and only differ in their fitting procedures, the two

approaches are expected to give similar results. Our logistic regression model slightly outperformed LDA likely because the LDA assumption that the observations are drawn from a Gaussian distribution with a common covariance matrix in each class fails to hold.

3.6 Linear and Quadratic Discriminant Analysis with All Terms From Sentiment Dictionary (Feature Engineered with PCA)

Similar to the logistic regression models, for our final LDA and QDA models, we also experimented with using all terms from the sentiment dictionary. PCA was implemented for feature engineering as well. The results are shown below:

Table 8: LDA (All Terms From Sentiment Dictionary with PCA) Test Results

	Precision	Recall	F1-Score	Support
Positive	0.86	0.82	0.84	12483
Negative	0.83	0.87	0.85	12517
Accuracy	–	–	**0.84568**	25000
Macro Avg	0.85	0.85	0.85	25000
Weighted Avg	0.85	0.85	0.85	25000

Computation Time: 0:00:15.433693

Table 9: QDA (All Terms From Sentiment Dictionary with PCA) Test Results

	Precision	Recall	F1-Score	Support
Positive	0.79	0.78	0.78	12483
Negative	0.78	0.79	0.79	12517
Accuracy	–	–	**0.78488**	25000
Macro Avg	0.78	0.78	0.78	25000
Weighted Avg	0.78	0.78	0.78	25000

Computation Time: 0:00:19.812929

For both the LDA and QDA models with features engineering using PCA, note that using the 2,000 terms from the original dictionary produced better results than using all terms from the sentiment dictionary. In particular, LDA using sentiment dictionary yielded an accuracy rate of around 0.023 lower than with our original dictionary. Similarly, QDA with the sentiment dictionary yielded an accuracy of around 0.037 lower than with the original dictionary.

3.7 K-Nearest Neighbors with Original Dictionary (Feature Engineered with PCA)

The K-Nearest Neighbors (KNN) algorithm is a non-parametric, supervised learning method used for both classification and regression tasks by calculating the distance between the data points and grouping together the points closest in proximity. To make a prediction about the class of an individual test data point, the algorithm finds the K number of points closest to the test data and then classifies the test data based on the majority classification of its neighboring points.

An advantage of KNN is that it is a highly flexible model that does not rely on the normality assumption. However, since we are working with a large dataset with high dimensionality, the computational costs are high as well.

To determine the optimal k value, we trained the model using five-fold cross validation and observed the following result:

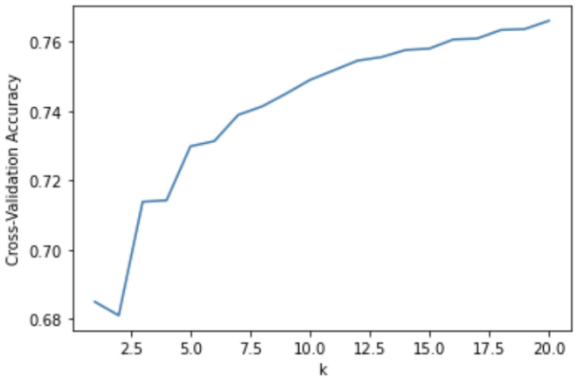

Figure 6: KNN 5-CV Result

Among all the accuracy values for each value of k, k = 19 has the highest accuracy of 75.879%. However, the computational costs of running a greater number of

k's would have been very high. Thus, we ultimately decided to use k = $\sqrt{n}$, as it is a common practice. After running a KNN with k = 157, we are presented with the following results:

Table 10: KNN (with Original Dictionary) Test Results, k = $\sqrt{n}$

	Precision	Recall	F1-Score	Support
Positive	0.90	0.43	0.58	12483
Negative	0.63	0.95	0.76	12517
Accuracy	–	–	**0.69204**	25000
Macro Avg	0.76	0.69	0.67	25000
Weighted Avg	0.76	0.69	0.67	25000

The results show that the KNN model seems to be able to classify positive reviews relatively well but has a tendency to classify negative reviews as positive. The validation set error rate generally tends to overestimate the actual test error rate; however, it appears that in this case, the model accuracy is about 6% less than the expected accuracy from the cross validation we conducted beforehand.

Although KNN is a highly flexible model and typically performs well on real-world datasets, it also has many limitations. For instance, the KNN algorithm tends to fall victim to the curse of dimensionality—that is, high-dimensional data worsens the performance of KNN. It is also very sensitive to the choice of k, so a different choice of k may produce better results than the ones presented here. Lastly, a huge drawback of KNN is the computational expense, especially for large datasets like the one we are working with. All of these limitations contribute to the poor performance of our KNN model.

4 Conclusion and Summary

4.1 Results and Analysis

By comparing the accuracy scores of each of our models on our testing dataset, we can effectively analyze the performance of our models. In Table 11 below, each model's accuracy score is displayed by lowest to highest score.

Table 11: Model Accuracy

Model	Dataset	Accuracy
Logistic Regression	Original Dictionary	**0.87360**
	Sentiment Dictionary	0.85440
	Original Dictionary (with PCA)	0.87324
	Sentiment Dictionary (with PCA)	0.85388
LDA	Original Dictionary (with PCA)	0.86864
	Sentiment Dictionary (with PCA)	0.84568
QDA	Original Dictionary (with PCA)	0.82172
	Sentiment Dictionary (with PCA)	0.78488
KNN	Original Dictionary (with PCA)	0.69204

According to the table above, we observe that logistic regression, particularly the one that uses the original dictionary with 2,000 terms, with an accuracy score of 0.8736, had the most accurate performance compared to our other models. Another logistic regression model that also uses 2,000 terms from the original dictionary, but feature-engineered with PCA, follows closely with an accuracy score of 0.8732. Logistic regression is easier to implement and interpret, and performs well when the data is linearly separable. It also makes no assumptions about the distribution of classes, which is beneficial to certain types of data where the distribution might be unknown.

Following the logistic regression model, our LDA model, with the original dictionary and PCA, was the second-most accurate model with an accuracy rate of 0.86864. It is possible that the sentiment lexicon data does not follow the normality assumption of LDA, so it is reasonable that logistic regression outperformed LDA slightly, but still has a comparable accuracy.

For all the models, we noticed that

our original dictionary consistently outperformed the pre-assembled sentiment dictionary, whether or not we did feature engineering using PCA. Intuitively, this makes sense because our dictionary was customized specifically for the context of movie reviews, while the sentiment dictionary contains the sentiment of commonly used words in any online context.

On the other end, our QDA model using the sentiment dictionary had a comparatively lower performance at an accuracy rate of 0.785. It is interesting to note that our KNN model performed with the least amount of accuracy at 69.2%. Since KNN and QDA models tend to be more flexible, we can assume that those two models may have overfit the training data, resulting in a lower accuracy score in terms of performance on the testing dataset.

To further determine the performance of our models, we compared how accurately each of the models predicted whether the sentiment was positive or negative. The precision, recall, and F1 scores for positive sentiments are presented below in table 12. These values for the negative sentiments are displayed in table 13 that follows later.

Table 12: Model "Positive" Classification Results

Model	Dataset	Positive Precision	Positive Recall	Positive F1-Score
Logistic Regression	Original Dictionary	0.88	0.86	0.87
	Sentiment Dictionary	0.87	0.84	0.85
	Original Dictionary (with PCA)	0.88	0.86	0.87
	Sentiment Dictionary (with PCA)	0.87	0.84	0.85
LDA	Original Dictionary (with PCA)	0.88	0.85	0.87
	Sentiment Dictionary (with PCA)	0.86	0.82	0.84
QDA	Original Dictionary (with PCA)	0.83	0.81	0.82
	Sentiment Dictionary (with PCA)	0.79	0.78	0.78
KNN	Original Dictionary (with PCA)	0.9	0.43	0.58

As can be seen in Table 12, the KNN model has the highest precision, but the lowest recall. This is likely due to an overfitting of the training data with the boundary around the training true positive sentiments, which leads to more precise predictions on which sentiments are labeled as positive in the testing dataset, and yet did not account for many of the true positives as well. Logistic regression and LDA with the original dictionary have the second highest precision at 0.88, which indicates that these models correctly predicted that a sentiment is positive when it is indeed positive 88% of the time. This suggests that these models captured a substantial portion of the true positive sentiments as a result of their linear boundaries from training. These two models also had the highest recall at 0.86 and 0.85, respectively.

With a precision of 0.79, QDA with the sentiment dictionary is the least precise in terms of predicting that a sentiment is positive when it is in fact positive. Likewise, QDA with the original dictionary has the second lowest precision score of 0.83, which means that QDA has the least precision out of all of our models overall.

Table 13: Model "Negative" Classification Results

Model	Dataset	Negative Precision	Negative Recall	Negative F1-Score
Logistic Regression	Original Dictionary	0.86	0.89	0.88
	Sentiment Dictionary	0.84	0.87	0.86
	Original Dictionary (with PCA)	0.86	0.89	0.88
	Sentiment Dictionary (with PCA)	0.84	0.87	0.86
LDA	Original Dictionary (with PCA)	0.85	0.89	0.87
	Sentiment Dictionary (with PCA)	0.83	0.87	0.85
QDA	Original Dictionary (with PCA)	0.82	0.83	0.82
	Sentiment Dictionary (with PCA)	0.78	0.79	0.79
KNN	Original Dictionary (with PCA)	0.63	0.95	0.76

Since the two classes in our dataset were balanced, we expect these metrics for the "Negative" class to be relatively the same as those of the "Positive" class.

Indeed, logistic regression using the original dictionary with and without PCA had the highest precision at 0.86 and high recall values at 0.89. Similarly, LDA using the original dictionary with PCA had the second highest precision at 0.85 and the highest recall at 0.89. Both logistic regression and LDA models are considered less flexi-

ble in comparison to QDA and KNN, meaning they have less variance and do not experience as much of an issue with overfitting. Thus, when making predictions on the testing dataset, these models had both the highest precision and highest recall.

On the other end, KNN had the lowest precision at 0.63 but the highest recall at 0.95. This is likely due to the additional flexibility of the KNN model in this case. It is possible that when training the model, it overfitted a decision boundary that ultimately did not work well with the testing dataset.

As a potential result of these flexibility issues, the distribution of F-1 scores for the "Negative" class is very similar to that of the other metrics. Logistic regression using the original dictionary with and without PCA achieved the highest F-1 score at 0.88, and LDA using the original dictionary with PCA achieved the second highest F-1 score at 0.87. In contrast, KNN attained the lowest F-1 score at 0.76.

Overall, we have observed that our logistic regression models using the original dictionary with and without PCA along with our LDA model using the original dictionary with PCA had the highest accuracy as well as the highest "Positive" and "Negative" precision, recall, and F-1 scores. Therefore, we conclude that using a model with less flexibility, such as logistic regression and LDA, would have the best performance in classifying a movie sentiment as either positive or negative from a written review.

4.2 Final Remarks

All of our models performed with an accuracy of above 75%, but there are still aspects of our process that allow for improvement.

Because our original dictionary outperformed the sentiment dictionary for every model, we suspect that our dictionary worked better because it is better suited for the context of our data. However, this could limit the generalizability of the dictionary for unknown data, since there are words that may be more commonly used for new movies that are not included in our dictionary. In the future, it may be possible to automate the entire princess of sentiment analysis for any context by developing an algorithm for generating a custom dictionary for any dataset.

Moreover, because the dataset contained so many records, we sometimes encountered difficulty training our models, especially for KNN, which sometimes took hours to run. It is possible to reduce computation time with some algorithmic improvement; though, ultimately, this limitation is not unique to this study but rather an illustration of current technological limitations.