IMDb Movie Review Sentiment Classification

Xinyu Dong
UID: 805568998

Zhuoya Li
UID: 105592223

Mingwei Wu
UID: 505571940

Bingyi Xu
UID: 705531203

Shukai Yin
UID: 105586364

Abstract

In this study, we explore binary classification for 50,000 IMDb movie reviews. We first employ data preprocessing techniques like text cleaning to work with the data. We then numericalize the text data with bag of words models (Count Vectorizer and TF-IDF) alongside sentiment lexicon analysis for the numerical conversion of reviews. Principal Component Analysis (PCA) is then applied for dimension reduction. Four machine learning models—K-Nearest Neighbors (KNN), Random Forest, Logistic Regression, and Support Vector Machine (SVM)— are trained and tested on both CountVectorizer and TF-IDF datasets. Logistic Regression and SVM show the highest accuracy, particularly with TF-IDF, demonstrating their effectiveness with large, sparse text datasets. Random Forest follow in terms of accuracy result; KNN underperfoms, especially with TF-IDF due to high-dimensional data challenges.

1 Introduction

The Internet Movie Database, commonly referred to as IMDb, is a digital repository that provides information about film actors, cinematic works, television series, and video games. [1] An example of IMDb movie review is demonstrated as follows, where the user gives text comments alongside a rating.


FEATURED REVIEW

★ 10/10

More Contemplative Than Others But Still Masterful

The best way to describe this movie is to say the entire film feels like the scene in Spirited Away where Chihiro is on the train going to Yubaba's sister's house. It feels nostalgic, contemplative, and yearning for a time when the world appeared to be easier (ie through the eyes of a young boy).

But, like all of Miyazaki's works, the film is filled with gorgeous animation and incredibly fantastical set pieces. It's a rare end of career film that feels like a filmmaker at the height of their powers. Don't go in expecting Spirited Away, Howl's Moving Castle or Ponyo - it's very different from all of those films even though it shares a lot of the same themes. Just know you're putting your trust in one of the best living filmmakers whose fire for telling stories is far from dimming.

Figure 1: An Example of IMDb Movie Review

This project focuses on the classification task of studying and predicting users' sentiments on a movie, whether positive or negative, based on their reviews. Our IMDb raw dataset [2] comprises 50,000 movie reviews (in text form) in total with clear labels of each review being considered positive or negative. The key challenges we addressed include the conversion of textual content into numerical format and the effective management of data with high-dimensional attributes using selected statistical models.

2 Data Preprocessing

In order to successfully classify whether the movie review is positive or negative, we first cleaned the dataset following a series of steps detailed below for preprocess and preparation for further analysis and modeling procedures.

First, we identified 418 duplicate reviews and removed them from the original dataset. We then removed unnecessary and redundant words by removing the html tags, expanding chatwords to clear contractions, cleaning emoticons, symbols, pictographs, map symbols, adding space after full stop, removing urls and punctuations, along with lemmatization [1] of words. With the cleaned text, we can further proceeded with our exploratory data analysis.

2.1 Descriptive Statistics

To gain a better understanding of the IMDb movie review dataset, we conducted some exploratory data analysis and visualized the results.

[1] Lemmatization is a text pre-processing technique used in natural language processing (NLP) models to break a word down to its root meaning to identify similarities.[3]

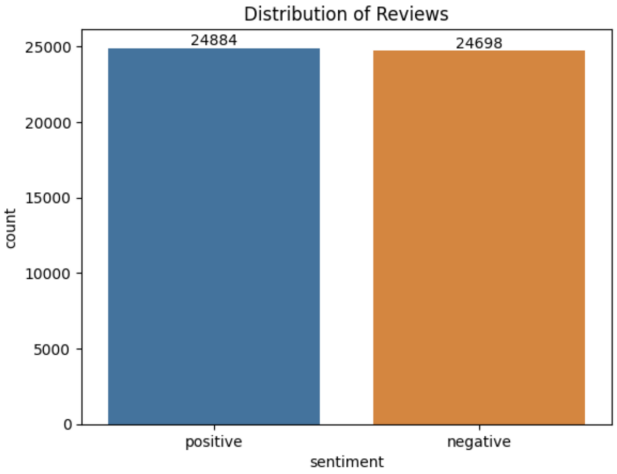

Figure 2: Distribution of Reviews

From the distribution, we can see that the dataset is basically balanced with almost equal number of positive and negative reviews. With the balanced dataset, accuracy should be considered as a reliable metric for evaluating model performance in our latter parts.

Figure 3: Word Cloud for Positive Reviews

We can see that if we only account for the frequency of the words appear in reviews, most of the words are not very useful in terms of classifying the review as positive or negative. In the positive review word cloud, despite some common words like "film," "movie," "one," and etc., there are a few words can directly points to the positive tone like "good," "well," or "love."

Figure 4: Word Cloud for Negative Reviews

Similar to the positive word cloud, the negative reviews include many "unuseful" words, and majority of the most frequent words overlap with the positive reviews. However, we can still observe words such as "bad" or "least," attributing to the reviews' attitude.

Overall, by merely comparing the frequency of words appearing in the reviews, we think it is difficult to distinguish the difference between the two classes, either positive or negative. Hence, we further carried out numericalization techniques in order to use machine learning models for classification.

2.2 Lexicon Sentiment Analysis and Numericalization

Classifiers and learning algorithms expect numerical feature vectors rather than raw text documents as presented in the original dataset. This is why we need to convert our movie review text into numerical vectors. Our approach is to adopt bag of words (BoW) since we care about the frequency of the words in text reviews while the order of words is irrelevant. Two common ways to represent bag of words are CountVectorizer and Term Frequency, Inverse Document Frequency (TF-IDF). Upon the bag of words model, we constructed a sentiment lexicon by importing a predefined dictionary classifying negative and positive words to vectorize all words in each review. Each word in the BoW model is matched with the lexicon[4]. If the word exists in the lexicon, its sentiment score is retrieved. Afterwards, as we applied two methods to numericalize the text data, we will further train different models on each of the two respectively.

2.2.1 Bag of Words - Count Vectorizer

In the previous part, we have successfully cleaned our text data, but for the modeling to work, it is necessary to convert this text-based data into numbers and a matrix called bag of words. Here we apply the CountVectorizer tool in sklearn package in python for this purpose [5].

The first step of Count Vectorizer is to tokenize the text documents, splitting them into individual words. By default, it considers tokens of two or more alphanumeric characters, filtering out single-character words. After tokenization, it builds a vocabulary of all unique tokens (words) from the entire set of documents. Each unique token gets an index assigned in this vocabulary. For each document, CountVectorizer creates a feature vector. The length of this vector is equal to the total number of unique words in the vocabulary. If a particular word is present in a document, the corresponding position in the feature vector is updated. Finally, the output of CountVectorizer is a sparse matrix representation of the document-term matrix [6]. This is efficient because most documents only contain a small subset of the vocabulary, leading to many zeros in the matrix.

2.2.2 Term Frequency and Inverse Document Frequency (TF-IDF)

TF-IDF, short for Term Frequency - Inverse Document Frequency, is a statistical method typically used in information retrieval and natural language processing. In CountVectorizer, the frequency of appearance of each word is used as a feature, which can give undue importance to rare words that might not be significant. TF-IDF mitigates this to some extent by reducing the weight of words that appear frequently across all documents. First proposed by Karen Spärck Jones as a statistical interpretation of term specificity [7], Inverse Document Frequency is a measure of how much information a given word or document provides. Mathematically, it is often defined as the logarithm inverse fraction of the documents that contain the word: $idf(t, D) = \frac{N}{|d \in D, t \in d|}$, where N refers to the total number of documents, and $|d \in D, t \in d|$ refers to the total number of documents in which our designated term t appears[8].

Term frequency, or TF for short, stands for the relative frequency of term t within document d. Though various ways exist for evaluating the term frequency of a specific term, we generally use $tf(t, d) = \frac{f_{t,d}}{T}$ to define the term frequency of term t in document d, where $f_{t,d}$ stands for the number of occurrences of term t in document d, and T means the total number of words in a given document d. Then, taking into account both the term frequency and the Inverse Document Frequency, we can calculate $tfidf(t, d, D)$ as $tf(t, d) \cdot idf(t, D)$[8].

Using the built in function TfidfVectorizer() in the machine learning library sklearn[8], we directly transformed the processed documents of text into a matrix of weights per document for each feature. At this point, our data is ready for feature engineering and further model fitting.

3 Feature Engineering: Dimension Reduction - PCA

To proceed with the modeling part, we first conducted the train-test split where training dataset consists of 75% of the dataset while the remaining is our testing dataset.

Since our data after the numericalization step potentially faces classification difficulties like overfitting and increased computational complexity due to the high dimensionality after lexicon sentiment analysis, we reduced the number of dimensions in our dataset using PCA, Principal Component Analysis[9]. As a popular machine learning technique to deal with high dimensional data, PCA finds a new coordinate system that is much lower in dimensions but contains most of the information stored in the original high dimension dataset. Given a target dimension r, PCA finds r linear combinations of the original features in the data set with each linear combination independent of the other. In order to determine the optimal target dimension r, we plot the cumulative variance explained by all of the principle components against the total number of principle components. Below are the plots for both the BOW - Count Vectorizer and the TFIDF Vectorizer:

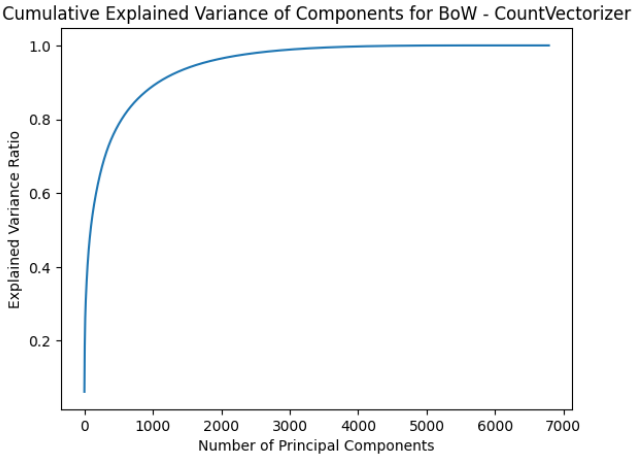

Figure 5: Cumulative Explained Variance of Components for BOW - Count Vectorizer

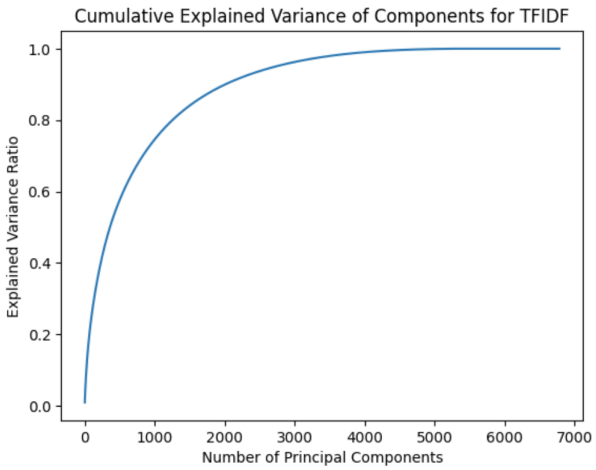

Figure 6: Cumulative Explained Variance of Components for TFIDF

As we can see from the above plots, for the BOW - Count Vectorizer numericalization our curve in the cumulative explained variance plot smooths out at roughly 3000 components. We decided to reduce our dimensions to 600 because at 600 Principal Components, more than 80% of the variation in the dataset can be explained by our Principal Components. Similarly, we observe that our curve in the cumulative explained variance plot smooths out at roughly 5000 components for TFIDF numericalization. We set our number of Components at 1500 to explain roughly 85% of the variation in our dataset.

4 Model Fitting

Having two datasets that were cleaned, numericalized using Count Vectorizer and TF-IDF respectively, and with dimensions reduced using PCA, we decided to fit four algorithms on each dataset in order to find the model in combination with the numericalization method that can best predict movie rating sentiments.

4.1 K-Nearest Neighbors

The K-Nearest Neighbor (KNN) machine learning model, when used for classification purposes, relies on the implicit assumption that data points closer in the feature space usually have similar labels. After specifying an odd integer value for K, the algorithm finds the K points in the training dataset that are closest to the input data point. It will then assign the most common label among the K neighbors as the predicted label for the input data point [10].

To determine the optimal value of K, we first applied 5-fold cross-validation on the two datasets. The accuracies are organized in the table below.

The dataset numericalized using Count Vectorizer showed higher accuracies over all values of k than the dataset numericalized using TF-IDF. While the accuracies are relatively similar within each numericalization method, K=11 yielded the highest accuracy of 72.1% for the count vectorizer method while K=3 yielded the highest accuracy of 57.9% for the TF-IDF method. Subsequently running the KNN models using these two optimal values yielded the following results:

K	Count Vectorizer	TF-IDF
3	0.7013	0.5787
5	0.7117	0.5785
7	0.7176	0.5517
9	0.7197	0.5410
11	0.7210	0.5535

Table 1: KNN 5-fold CV Accuracies

	Precision	Recall	F1-Score	Support
Negative	0.78	0.59	0.68	6152
Positive	0.68	0.84	0.75	6244
Accuracy	-	-	0.72	12396
Macro Avg	0.73	0.72	0.71	12396
Weighted Avg	0.73	0.72	0.71	12396

Table 2: KNN Test Results (BoW - Count Vectorizer)

	Precision	Recall	F1-Score	Support
Negative	0.55	0.88	0.67	6152
Positive	0.70	0.28	0.40	6244
Accuracy	-	-	0.58	12396
Macro Avg	0.62	0.58	0.54	12396
Weighted Avg	0.62	0.58	0.54	12396

Table 3: KNN Test Results (TF-IDF)

The tables show quite different results for datasets numericalized using Count Vectorizer and TF-IDF. For the Count Vectorizer method, KNN has a higher precision of 78% when predicting negative movie ratings, while the TF-IDF method yields a higher precision of 70% when predicting positive movie ratings. The overall accuracies for both methods are approximately the same as the expected accuracies yielded from cross validation. The Count Vectorizer method results in a higher accuracy of 72% comapred to the TF-IDF method accuracy of only 58%

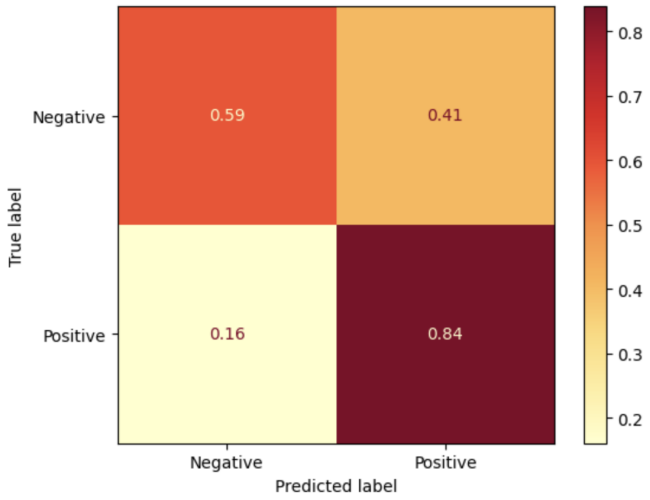

Figure 7: Confusion Matrix of KNN (BoW - Count Vectorizer)

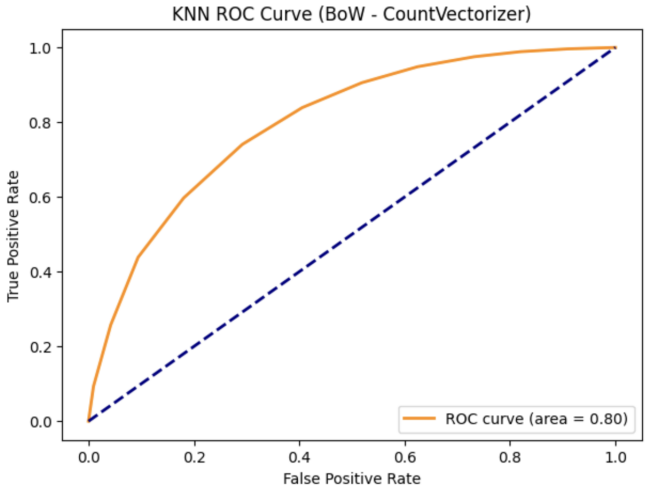

Figure 9: ROC Curve of KNN (BoW - Count Vectorizer)

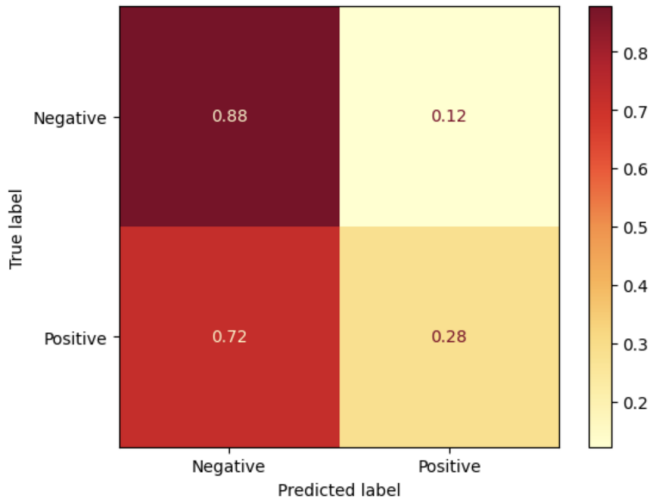

Figure 8: Confusion Matrix of KNN (TF-IDF)

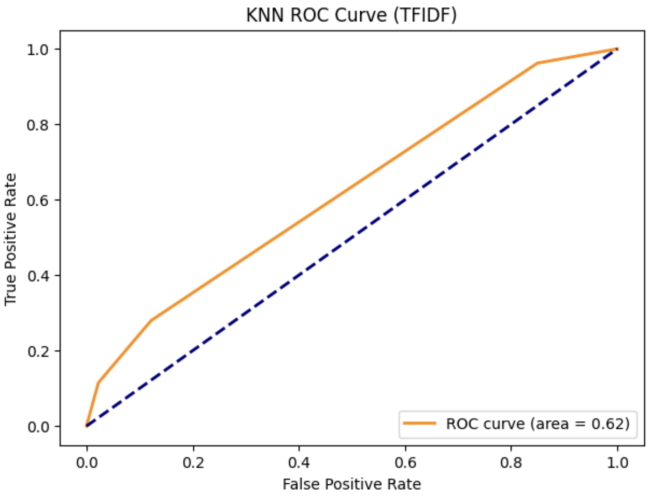

Figure 10: ROC Curve off KNN (TF-IDF)

The confusion matrices show that both KNN models' performances are not the most ideal, especially since the Count Vectorizer method has a low true negative rate of 0.59 and the TF-IDF method has a very small true positive rate of only 0.28.

The Count Vectorizer method yields an AUC value of 0.8, suggesting that the quality of the KNN algorithm when fitted on this dataset is better than the TF-IDF method, where the AUC value is only 0.62. It is also worth noting that KNN took the longest time to run among all four models. As both datasets have relatively high dimensionality even after PCA, it is computationally expensive to calculate all the Euclidean distances between the training data and the input data, resulting in inefficiencies and potentially worsened performances. Since these results imply that the KNN model quality is merely good or satisfactory but not excellent, we decided to fit several other models to find ones that might perform better.

4.2 Random Forest

The Random Forest algorithm combines the outputs of multiple decision trees. As a non-parametric supervised learning algorithm, decision tree aims to predict the label of the input data point by applying simple decision rules concluded from interpreting the training dataset features. While decision trees are relatively easy to interpret, it is prone to overfitting, especially on data with a large number of features, as the tree can continuously split and generate new nodes until each subset only contains samples from one class [11]. On the other hand, Random Forest is able to reduce overfitting by averaging multiple decision tree outputs and is known for handling large and complex datasets.

In a Random Forest model, a subset of data points and features are selected from the training dataset to construct each decision tree. Since we are using Random Forest for classification here, the final output will be decided based on majority voting of the ouputs generated by each decision tree [12]. For instance, if we have 5 decision trees and 3 of them output negative ratings, the eventual Random Forest Classifier prediction of the input data will be negative.

Following a similar procedure, we fitted the Random Forest models on both sets of data and obtained the following results.

	Precision	Recall	F1-Score	Support
Negative	0.80	0.76	0.78	6152
Positive	0.77	0.81	0.79	6244
Accuracy	-	-	0.79	12396
Macro Avg	0.79	0.79	0.79	12396
Weighted Avg	0.79	0.79	0.79	12396

Table 4: Random Forest Test Results (BOW - Count Vectorizer)

	Precision	Recall	F1-Score	Support
Negative	0.80	0.79	0.80	6152
Positive	0.80	0.81	0.80	6244
Accuracy	-	-	0.80	12396
Macro Avg	0.80	0.80	0.80	12396
Weighted Avg	0.80	0.80	0.80	12396

Table 5: Random Forest Test Results (TF-IDF)

The Random Forest Classifier model results for both the Count Vectorizer and TF-IDF methods show significant improvements over the KNN methods. The precision for predicting positive and negative ratings are all equal to 80% except the precision falls slightly to 77% when classifying positive ratings for the Count Vectorizer dataset. The recall and F1-scores are also in the same range of around 80%.

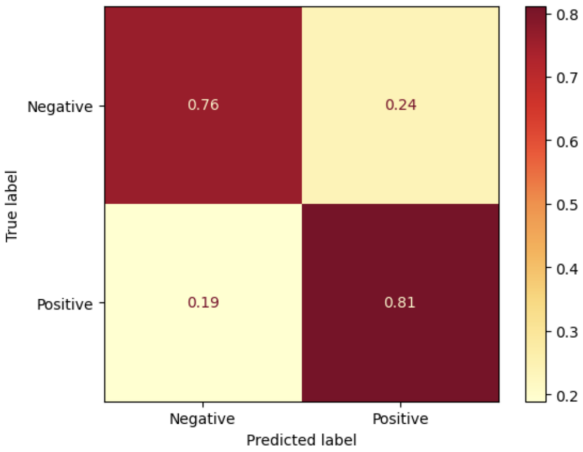

Figure 11: Confusion Matrix of Random Forest (BoW - Count Vectorizer)

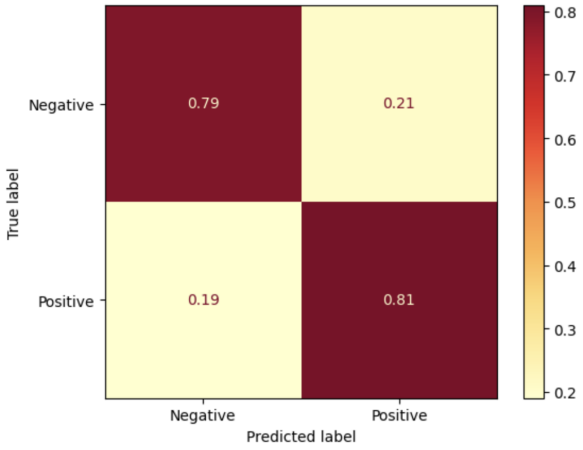

Figure 12: Confusion Matrix of Random Forest (TF-IDF)

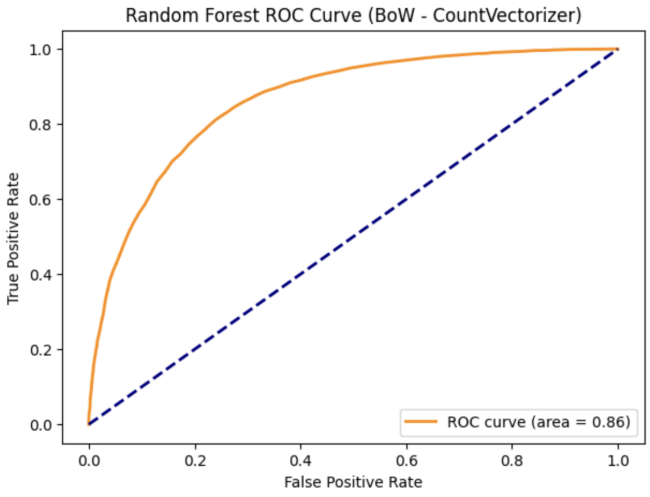

Figure 13: ROC Curve of Random Forest (BoW - Count Vectorizer)

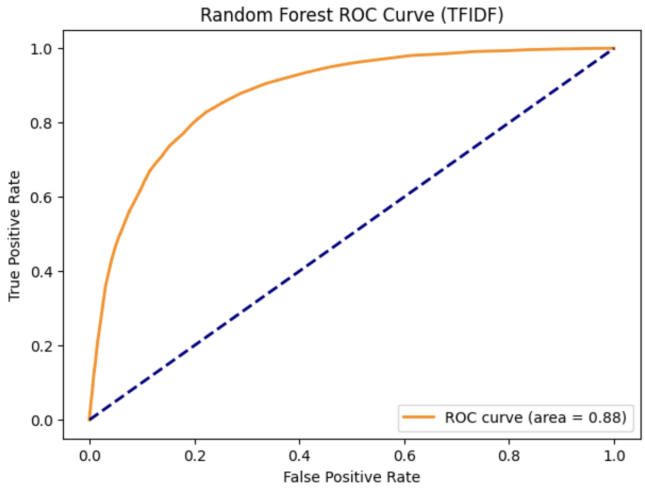

Figure 14: ROC Curve of Random Forest (TF-IDF)

The AUC values for both Random Forest models are 0.86 and 0.88 respectively, implying that the test quality are relatively good when trained on both datasets.

4.3 Logistic Regression

Logistic Regression estimates the conditional probability $P(\tilde{Y} = 1|\mathbf{X} = \mathbf{x})$, discerning the likelihood of a binary event's occurrence based the data. It employs a linear decision boundary and minimizing logistic loss to determine these probabilities. [13]

To address the potential issue of over-fitting, we incorporated L2 regularization into our logistic regression model. This regularization approach effectively limits the size of the coefficients, promoting model generalization while retaining all input features. This is especially relevant since our feature set had already been reduced through PCA.

The following results are obtained from runing logistic regression on dataset pre-processed using Count Vectorizer and TF-IDF Vectorizer respectively:

	Precision	Recall	F1-Score	Support
Negative	0.86	0.82	0.84	6152
Positive	0.83	0.87	0.85	6244
Accuracy	-	-	0.84	12396
Macro Avg	0.84	0.84	0.84	12396
Weighted Avg	0.84	0.84	0.84	12396

Table 6: Logistic Regression Test Results (BoW - Count Vectorizer)

	Precision	Recall	F1-Score	Support
Negative	0.86	0.83	0.85	6152
Positive	0.84	0.87	0.85	6244
Accuracy	-	-	0.85	12396
Macro Avg	0.85	0.85	0.85	12396
Weighted Avg	0.85	0.85	0.85	12396

Table 7: Logistic Regression Test Results (TF-IDF)

Logistic regression gives higher precision in predicting negative sentiment than positive in both datasets. However, the precision values for either type are similar when compared across the two datasets, with slightly improved result on TF-IDF dataset by 1% on positive predictions.

In contrast, recall figures shows that more positive sentiments are correctly predicted among all the correct predictions. TF-IDF dataset again resulted in a marginally better predicting result compared with Count Vectorizer on negative predictions.

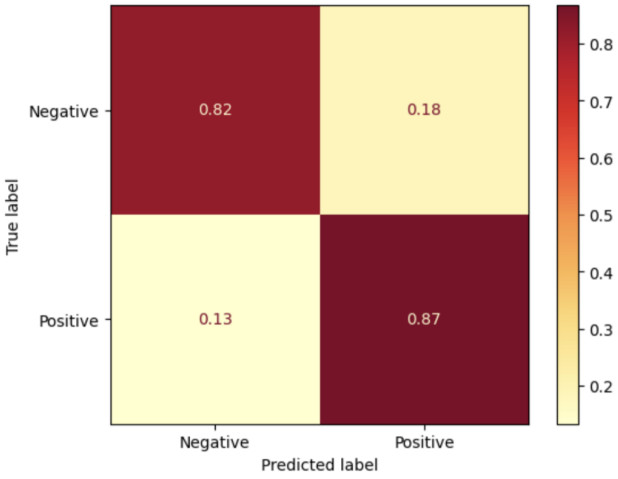

Figure 15: Confusion Matrix of Logistic Regression(BoW - CountVectorizer)

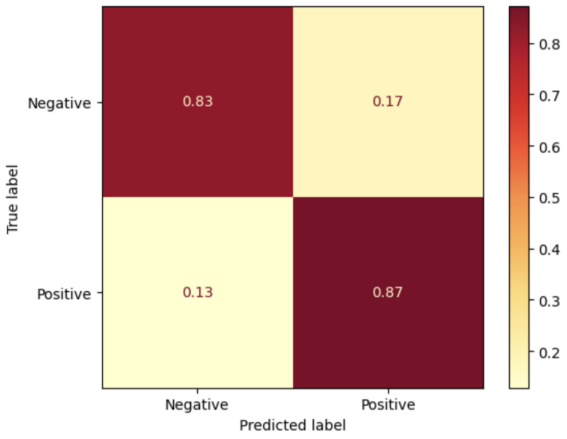

Figure 16: Confusion Matrix of Logistic Regression (TF-IDF))

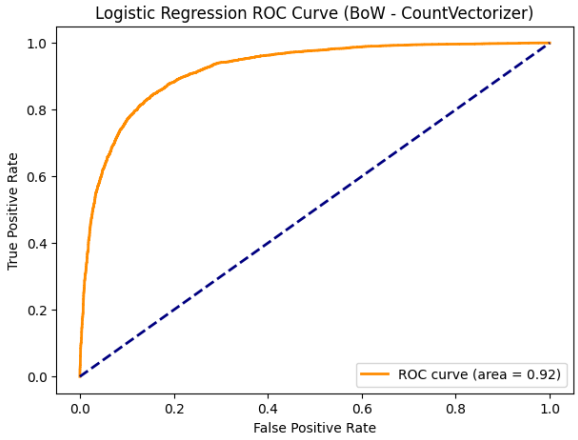

Figure 17: ROC Curve for Logistic Regression (BoW - CountVectorizer)

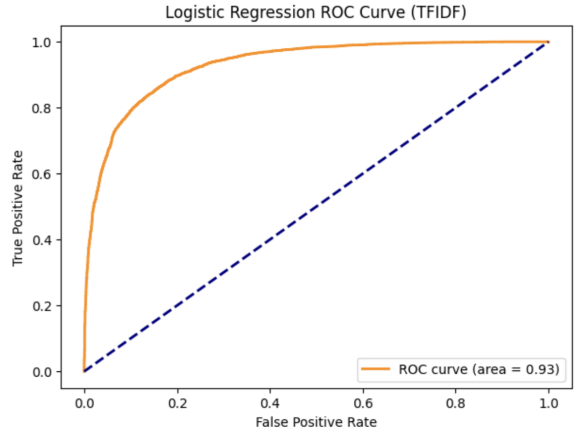

Figure 18: ROC Curve for Logistic Regression (TF-IDF)

The Count Vectorizer dataset yields an accuracy of about 84.3%, with a modestly higher recall for the positive class and an AUC of 0.92, indicating good class differentiation. The model tends more towards false positives than false negatives. For the TF-IDF vectorizer dataset, accuracy is slightly higher at approximately 84.9%, with improved positive class recall and a marginally better AUC of 0.93. In summary, logistic regression are strong in prediction for both datasets, with the TF-IDF method displaying minimally superior accuracy and class discrimination.

4.4 Support Vector Machine

The Support Vector Machine (SVM) algorithm seeks to identify an optimal decision hyperplane that maximizes the margin between different class data points. In contrast to logistic regression, which outputs probabilities, SVM aims for the most decisive class separation. We adopted a linear SVM (linear classifier with SGD training) to find a hyperplane in the feature space to maximize the margin, which is the distance between the hyperplane and the nearest data points from each class [14].

Accounting for computational efficiency, we chose to use the Stochastic Gradient Descent modeling technique with hinge loss function, which is known as SVM, rather than svm.SVC from the sklearn package. The method we use only takes a few seconds while svc would take up to ten minutes.

The subsequent outcomes are derived from applying an linear SVM on datasets pre-processed through Count Vectorizer and TF-IDF Vectorizer:

	Precision	**Recall**	**F1-Score**	**Support**
Negative	0.86	0.82	0.84	6152
Positive	0.83	0.86	0.85	6244
Accuracy	-	-	0.84	12396
Macro Avg	0.84	0.84	0.84	12396
Weighted Avg	0.84	0.84	0.84	12396

Table 8: Linear SVM Test Results (BoW - Count Vectorizer)

	Precision	**Recall**	**F1-Score**	**Support**
Negative	0.85	0.84	0.85	6152
Positive	0.85	0.86	0.85	6244
Accuracy	-	-	0.85	12396
Macro Avg	0.85	0.85	0.85	12396
Weighted Avg	0.85	0.85	0.85	12396

Table 9: Linear SVM Test Results (TF-IDF)

Linear SVM yields high precision of 86% and 85% when classifying negative sentiments in Count Vectorizer and TF-IDF datasets. While the model give lower precision score for positive predictions for Count Vectorizer, there is no obvious difference between postive and negative class for TF-IDF. The F1-Scores hover around 85% for both classes in either dataset.

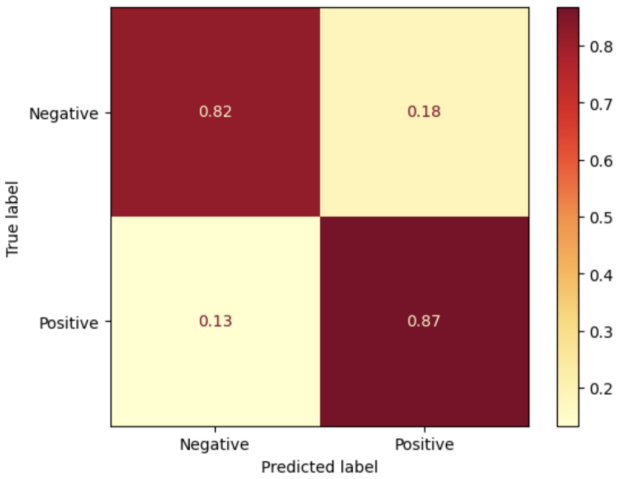

Figure 19: Confusion Matrix for SVM (BoW - CountVectorizer)

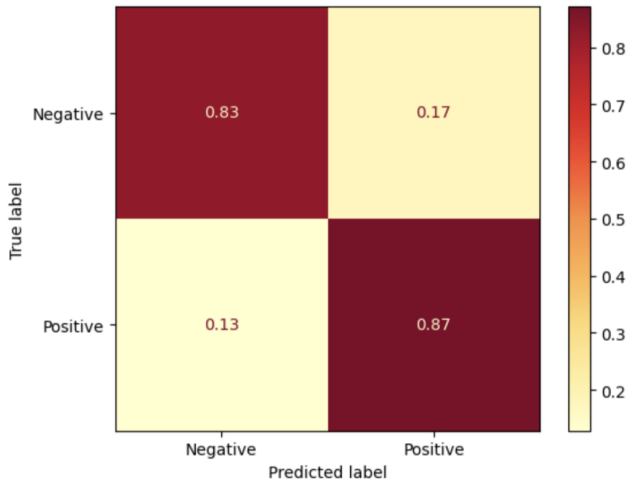

Figure 20: Confusion Matrix for SVM (TFIDF)

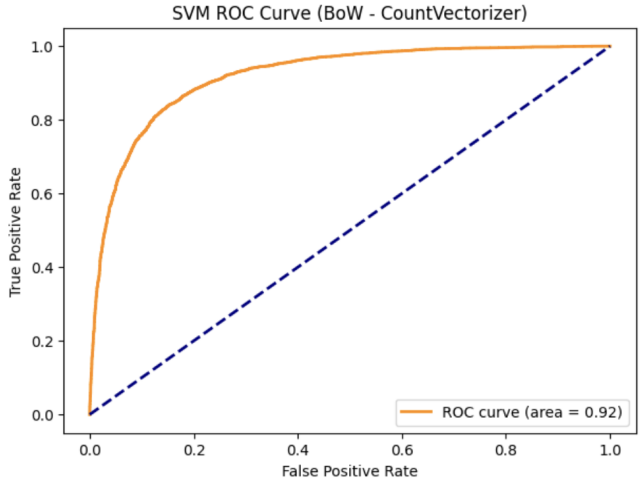

Figure 21: ROC Curve of SVM (BoW - Count Vectorizer)

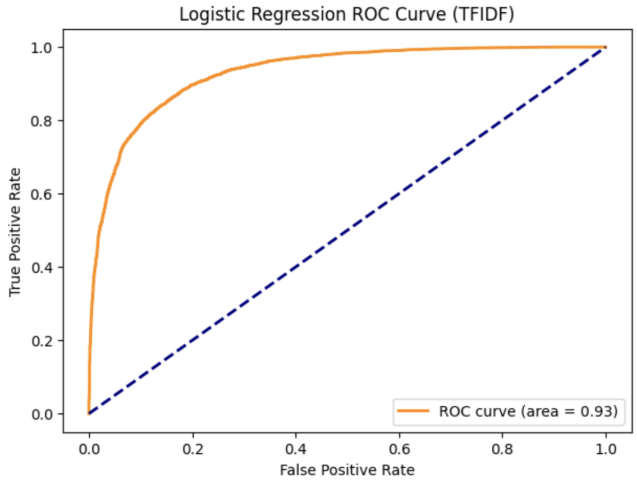

Figure 22: ROC Curve of SVM (TF-IDF)

The Count Vectorizer dataset yields true negative and positive rates of 0.82 and 0.86, respectively, with an AUC of 0.92, denoting effective class separation. The TF-IDF dataset has slightly better result in model's discernment, demonstrated by an improved true negative rate of 0.84 and an AUC of 0.93, while having the same true positive rate. Thus, the linear SVM demonstrates robust classification capabilities with both methodologies, with TF-IDF achieving marginally superior performance.

5 Conclusion and Summary

5.1 Results and Analysis

We start evaluating our models by comparing their accuracy scores on the test dataset, arranging them from the highest to the lowest.

Model	Accuracy (Count Vectorizer)	Accuracy (TF-IDF)
LR	0.8430	0.8499
SVM	0.8425	0.8510
RF	0.7876	0.8015
KNN	0.7177	0.5768

Table 10: Comparison of Model Accuracies

From the table, we observe our Logistic Regression and Support Vector Machine provides the most accurate prediction in both cases. LR and SVM especially works well with the large, sparse text datasets, such as dealing with feature spaces where the number of features (i.e. words) is greater than the number of samples.

Random Forest offers the next best prediction accuracy across both datasets. Although Random Forest (RF) is a strong model, it may not perform as effectively in high-dimensional spaces without precise tuning. Additionally, it usually takes longer to train than Logistic Regression and Support Vector Machine, which could impact performance if the models weren't allowed sufficient time to converge.

KNN gives the lowest accuracy score in both scenarios. One reason could be that KNN is not as powerful working with high-dimensional data due to the "curse of dimensionality."

On the other hand, comparing the model performance for data processed with Count Vectorizer and TF-IDF Vectorizer shows the following findings:

1. Overall, model performance is better at TF-IDF vectorized dataset

2. The discrepancy in accuracy score is the smallest for SVM

3. KNN shows an interesting deteriorated performance for TF-IDF dataset

TF-IDF theoretically captures more indicative features in documents, leading to better predicting power of models. SVM's accuracy is less improved as its effectiveness in finding the optimal hyperplane is less sensitive to the scale of input features. However, models like KNN are more affected by the relative weighting of features, potentially explaining why KNN has a large drop in performance with TF-IDF.

Diving deeper into the specific metrics and performances on predicting the positive and negative movie ratings respectively, we organized the four models' precision, recall, and F1-scores for clearer display and comparisons.

Model	Positive Precision	Positive Recall	Positive F1-Score
LR	0.83	0.87	0.85
SVM	0.83	0.86	0.85
RF	0.77	0.81	0.79
KNN	0.68	0.84	0.75

Table 11: Comparison of Model Performances when Classifying Positive Movie Ratings (BoW - Count Vectorizer)

Model	Positive Precision	Positive Recall	Positive F1-Score
LR	0.84	0.87	0.85
SVM	0.85	0.86	0.85
RF	0.80	0.81	0.80
KNN	0.70	0.28	0.40

Table 12: Comparison of Model Performances when Classifying Positive Movie Ratings (TF-IDF)

Model	Negative Precision	Negative Recall	Negative F1-Score
LR	0.86	0.82	0.84
SVM	0.86	0.82	0.84
RF	0.80	0.76	0.78
KNN	0.78	0.59	0.68

Table 13: Comparison of Model Performances when Classifying Negative Movie Ratings (BoW - Count Vectorizer)

Model	Negative Precision	Negative Recall	Negative F1-Score
LR	0.86	0.83	0.85
SVM	0.85	0.84	0.85
RF	0.80	0.79	0.80
KNN	0.55	0.88	0.67

Table 14: Comparison of Model Performances when Classifying Negative Movie Ratings (TF-IDF)

Again, from the above tables, we can see that the performances of Logistic Regression, SVM, and Random Forest are ideal in both cases (BoW with Count Vectorizer and with TF-IDF Vectorizer). The performance of KNN is significantly less optimal when applied to text data that has been transformed using the Bag of Words model with TF-IDF vectorization. KNN works poorly with high-dimensional data, and text data, especially after being transformed into BoW with TF-IDF, usually results in a very high-dimensional space. Even though we have applied PCA on top of TF-IDF, the dimension of our text data is still relatively high. KNN does not perform well in high-dimensional spaces because the distance between most pairs of points becomes almost equal, making it hard to find close neighbors that are truly similar. Moreover, the vectors resulting from TF-IDF vectorization are typically very sparse with many entries being zero. In such sparse spaces, distance metrics (like Euclidean distance) often become less meaningful, which negatively impacts the performance of KNN. As shown in the results, these two drawbacks of KNN come into play for both Count Vectorizer and TF-IDF while the latter being more heavily impacted.

5.2 Final Remarks

Although the overall performances of our models is relatively high with a mean around 80% accuracy, there are some inherent limitations throughout the methods we used and some room for improvements in the future.

For the data itself, sometimes reviews might be neutral, and if these are forced into a positive or negative classification, it might lead to inaccuracies in analysis. There also exists some inherent limitations to our data processing step, particularly the sentiment analysis part. In terms of sentiment analysis, as we imported the pre-made positive and negative word dictionaries, we neglected the context the words appear in each sentence. It doesn't account for word order, negations, or context, which can lead to inaccuracies. Also, detecting sarcasm or irony in text is a challenge for sentiment analysis models and can lead to misclassification. Advanced techniques like using n-grams can overcome such limitation to certain extent. For our numericalization method, bag of words with CountVectorizer and TF-IDF vectorizer, BoW often times treats text as a 'bag' of words, meaning it disregards the order in which words appear. This loss of context can lead to a lack of understanding of the meaning conveyed by word order, which is often crucial in natural language processing. For PCA, since BoW with CountVectorizer often results in a sparse matrix, PCA is typically more effective on dense data rather than sparse data. For sparse data, techniques like Truncated Singular Value Decomposition (SVD) might be more appropriate.

In the future, for feature engineering, to achieve richer feature representations, we can apply some advanced NLP techniques like word embeddings like Word2Vec, GloVe or contextual embeddings like BERT. Moreover, our current project applied train-test split which face several biases, so implementing K-fold cross validation can provide a better assessment of model performance across different subsets of the

data. For model optimization, we would potentially include hyperparameter tuning for each model (we already tuned for KNN) as systematic approaches like grid search or random search can be used for optimization.

References

[1] IMDb *Wikipedia*, 2017 https://en.wikipedia.org/wiki/IMDb, Accessed 01-December-2023.

[2] IMDb Dataset of 50K Movie Reviews. *Kaggle*, 2018. https://www.kaggle.com/datasets/lakshmi25npathi/IMDb-dataset-of-50k-movie-reviews, Accessed 01-December-2023.

[3] A. Gillis. Definition of Lemmatization. *TechTarget*, 2023. https://www.techtarget.com/searchenterpriseai/definition/lemmatization#:~:text=Lemmatization%20is%20the%20process%20of,processing%20(NLP)%20and%20chatbots., Accessed 01-December-2023.

[4] Opinion Lexicon. *Kaggle*, 2017. https://www.kaggle.com/datasets/nltkdata/opinion-lexicon, Accessed 01-December-2023.

[5] CountVectorizer. *ScikitLearn*, 2023. https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html, Accessed 02-December-2023.

[6] What Is CountVectorizer In NLP. *PianalytiX*, n.d. https://pianalytix.com/countvectorizer-in-nlp/#:~:text=CountVectorizer%20means%20breaking%20down%20a,data%20needs%20to%20be%20vectorized., Accessed 02-December-2023.

[7] K.S. Jones. IDF term weighting and IR research lessons. *Computer Laboratory, University of Cambridge*, 2004. https://www.cl.cam.ac.uk/archive/ksj21/ksjdigipapers/jdoc04.pdf, Accessed 03-December-2023.

[8] F. Karabiber. TF-IDF — Term Frequency-Inverse Document Frequency. *Fatih Karabiber*, 2023. https://www.learndatasci.com/glossary/tf-idf-term-frequency-inverse-document-frequency/#:~:text=Term%20Frequency%20%2D%20Inverse%20Document%20Frequency%20(TF%2DIDF)%20is,%2C%20relative%20to%20a%20corpus)., Accessed 03-December-2023.

[9] A. Kumar. PCA explained variance concepts with python example. *Vitalflux*, 2022. https://vitalflux.com/pca-explained-variance-concept-python-example/, Accessed 03-December-2023.

[10] A Complete Guide to K-Nearest Neighbors (Updated 2023). *Analytics Vidhya*, 2023. https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/, Accessed 03-December-2023.

[11] Decision Trees. *ScikitLearn*, 2023. https://scikit-learn.org/stable/modules/tree.html, Accessed 03-December-2023.

[12] Understand Random Forest Algorithms With Examples (Updated 2023). *Analytics Vidhya*, 2023. https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/#Difference_Between_Decision_Tree_and_Random_Forest, Accessed 05-December-2023.

[13] Logistic Regression *ScienceDirect*, 2007 https://www.sciencedirect.com/topics/computer-science/logistic-regression#:~:text=Logistic%20regression%20is%20a%20process,%2Fno%2C%20and%20so%20on., Accessed 05-December-2023.

[14] Linear Support Vector Machine *ScienceDirect*, 2017 https://www.sciencedirect.com/topics/engineering/linear-support-vector-machine, Accessed 06-December-2023.