IMDB Review Sentiment Prediction

Fengruo Zhang, Maocheng Xiong, Yuge Song, Zekai Han

Dec 15, 2023

Abstract

This study revolves around the exploration and analysis of the IMDB Dataset of 50,000 movie reviews, designed for binary sentiment classification. The primary objective is to analyze the relationship between the textual content of reviews and the sentiments expressed. Using a diverse array of models, including Logistic Regression, K-Nearest Neighbors, Linear and Quadratic Discriminant Analysis, Random Forests, Gaussian Naive Bayes, and Support Vector Machine, we aim to uncover patterns and nuances within the language of movie reviews.

1 Introduction

IMDB, or the Internet Movie Database, is one of the most popular and extensive sources of entertainment-related content on the Internet. It provides information about films, television series, video games, and the people involved in their production. On IMDB, users can rate movies and TV shows, and they can write and read reviews, providing insights and opinions about the content. The primary goal of this project is to construct statistical models capable of predicting ratings based on the content and sentiment expressed within the reviews. Ultimately, we will decide which model has the highest predicting accuracy.

2 Data Pre-processing

In the beginning, We perform data pre-processing for the preparation of applying our models.

2.1 Data Description

Our dataset comprises a collection of 50,000 movie reviews, with each review accompanied by a few sentences and a corresponding sentiment indicator, showing whether the review is positive or negative. Additionally, we are given two separate files, each containing positive sentiment words and negative sentiment words, respectively.

2.2 Data Cleaning

To address formatting issues within the reviews, the initial step involves the removal of special characters, such as brackets and parentheses, by using regular expressions. Additionally, we eliminate extra spaces between words for the uniformity of the textual data. To standardize the dataset further, we convert all words to lowercase. Finally, lemmatization is applied to the reviews, so we can reduce the words to their root forms. This method ensures a comprehensive transformation, exemplified by instances where words such as "running" are lemmatized to "run," and "better" is reduced to "good. (Khyani et al. 2021)" By implementing the above multi-step preprocessing approach, the reviews are refined to a standardized format for data vectorization.

2.3 Data Vectorization

We begin this process by reading both positive and negative files containing sentiment-specific words, and then we amalgamate them into a single vocabulary list. Subsequently, we create a vectorizer using the CountVectorizer from the scikit-learn library. This transformative process involves converting the textual content of reviews within our dataset into a numerical matrix denoted as X, with several parameters being set. Specifically, we set the threshold of the max features to be 1500 to avoid overfitting. Then, we only consider important but not abusively use words by setting the document frequency between 0.1 and 0.7 inclusively. Further, leveraging the Natural Language Toolkit library, we exclude common English stop words, such as "and," "the," and "is", which are considered to contain minimal meaningful information.

Finally, we convert sentiment labels from string to integer datatype. If the sentiment is positive, the corresponding element in y is set to 1; if the sentiment is negative, it is set to 0.

2.4 Principal Component Analysis

PCA is a popular method for reducing the dimension of a dataset because it can be applied to any data matrix to capture the structure of data (Wold et al., 1987). So we reduce the dimension of our dataset by principal component analysis(PCA) to get rid of the curse of dimensionality.

Firstly, we apply a hyperparameter tuning process to select the optimal value for the $n\_components$ of PCA. We get the information about the explained variance and plot the cumulative variance as shown in Figure 1.

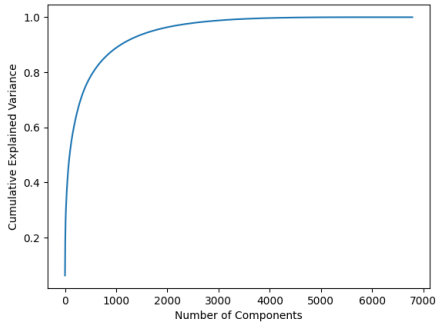

Figure 1: Cumulative Explained Variance vs. number of components

We select $n\_components$ to be 1100, which explains 90% variance. Therefore, the dimension of our new dataset, $reduced\_X$, is 50000×1100 now.

2.5 Data Splitting

Before training models, we randomly divide the dataset, both $reduced\_X$ and y, into training and testing sets in an 8:2 ratio to test our model performances.

2.6 Hardware Accelerator

We utilize a hardware accelerator, the T4 GPU, to accelerate the execution of algorithms including K-nearest neighbors and support vector machine. After applying principal component analysis to reduce the dimension of the dataset, we still need to treat a dataset with the dimension of 50000×1100. Therefore, we have to overcome the unexpected computational challenges. T4 GPU, allowing performing calculations simultaneously because of parallel processing, increases the efficiency of implementing K-nearest neighbors and support vector machine to our dataset.

3 Experiment

3.1 Logistic Regression

Logistic regression, a supervised machine learning algorithm, is primarily employed for classification tasks. Its objective is to predict the probability of an instance belonging to a specific class. Despite its name, logistic regression is used in classification rather than regression. It utilizes the linear regression function's output as input and applies a sigmoid function to estimate the probability for the designated class. This approach is instrumental in predicting categorical dependent variables based on a set of independent variables.

For our model, we also add the $l2$ penalty. In this way, we can prevent a large coefficient to avoid overfitting. Meanwhile, Ridge Regression ensures computational advantages because it only fits a single model(Pereira et al. 2015).

After training the Logistic Regression model, we generate the following results from our testing dataset:

	precision	recall	f1-score	support
0	0.87	0.84	0.85	4957
1	0.84	0.87	0.86	5043
accuracy	-	-	0.85	10000
macro avg	0.86	0.85	0.85	10000
weighted avg	0.86	0.85	0.85	10000

Figure 2: Relative metrics of logistic regression

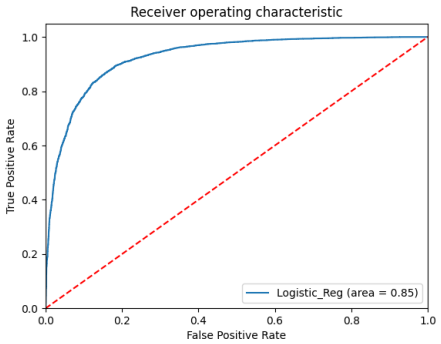

Figure 3: ROC of logistic regression

We can see that this model fits well to both true and false categories, and has a high accuracy.

3.2 Gaussian Naive Bayes

Gaussian Naive Bayes (GNB) is a classification technique relying on a probabilistic approach and Gaussian distribution. This method assumes that each parameter (features or predictors) independently contributes to predicting the output variable. The collective predictions of all parameters yield a probability for the dependent variable's classification into each group. The ultimate classification is assigned to the group with the highest probability (Martins, 2023).

After applying this method, we have results as follows:

	precision	recall	f1-score	support
0	0.65	0.78	0.71	4957
1	0.73	0.58	0.65	5043
accuracy	-	-	0.68	10000
macro avg	0.69	0.68	0.68	10000
weighted avg	0.69	0.68	0.68	10000

Figure 4: Relative metrics of Gaussian Naive Bayes

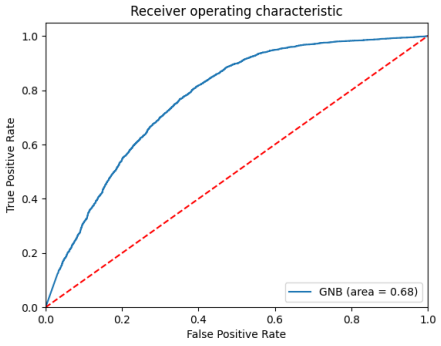

Figure 5: ROC of Gaussian Naive Bayes

The performance of this model is not satisfactory, because low precision and recall imply that this model does not return enough relevant true results. In other words, the accuracy of this model is small.

3.3 Random Forests

A random forest serves as a machine learning method employed for addressing both regression and classification challenges. It employs ensemble learning, a technique that amalgamates multiple classifiers to address intricate problems.

Comprising numerous decision trees, a random forest algorithm creates a 'forest' that undergoes training via bagging or bootstrap aggregating. Bagging, recognized as an ensemble meta-algorithm, enhances the accuracy of machine learning algorithms.

The algorithm determines its outcome based on the predictions of these decision trees, averaging or taking the mean of their outputs. Enhancing precision is achieved by increasing the number of trees.

In contrast to a decision tree algorithm, a random forest overcomes its limitations. It mitigates overfitting issues in datasets, thereby improving precision. Moreover, it generates predictions with minimal configuration requirements in packages(Mbaabu, 2020).

For our model, after fitting the randomized search cross-validation to our dataset, we get the best parameters set. We choose 15 trees in the forest, 10 maximum depth of the tree, 5 minimum samples required to split an internal node, 3 minimal samples required to be at a leaf node, and using the whole dataset to build each tree. Meanwhile, we utilize the Gini impurity function to measure the quality of a split. Our results are as follows:

	precision	recall	f1-score	support
0	0.80	0.72	0.76	4957
1	0.74	0.82	0.78	5043
accuracy	-	-	0.77	10000
macro avg	0.77	0.77	0.77	10000
weighted avg	0.77	0.77	0.77	10000

Figure 6: Relative metrics of Random Forest

From these results, we can find that this model's precision and recall show an imbalance in the results. So this model may not fit our data well.

3.4 Linear and Quadratic Discriminant Analysis

Linear Discriminant Analysis (LDA) and Quadratic Discriminant Analysis (QDA) are classic classifiers, featuring linear and quadratic decision surfaces, respectively. Known for their closed-form solutions, inherent multiclass support, practical effectiveness, and absence of hyperparameters, they provide versatile classification.

LDA offers supervised dimensionality reduction, projecting input data onto a linear subspace for optimal class separation. This strong dimensionality reduction is particularly meaningful in multiclass scenarios.

For LDA, in our cases, we assume the data are drawn from a Gaussian distribution with a common covariance matrix in each class. Also, we choose the solver to be Singular Value Decomposition, which does not compute the covariance matrix and can perform better for our dataset with a large dimension. The only difference QDA has is that it does not assume the same covariance matrix in each class.

Our results based on these two methods are:

	precision	recall	f1-score	support
0	0.87	0.81	0.84	4957
1	0.83	0.88	0.85	5043
accuracy	-	-	0.85	10000
macro avg	0.85	0.85	0.85	10000
weighted avg	0.85	0.85	0.85	10000

Figure 7: Relative metrics of Linear Discriminant Analysis

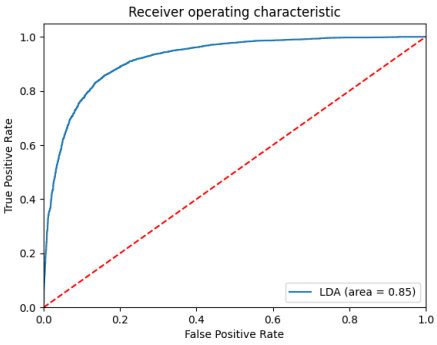

Figure 8: ROC of Linear Discriminant Analysis

	precision	recall	f1-score	support
0	0.79	0.81	0.80	4957
1	0.81	0.79	0.80	5043
accuracy	-	-	0.80	10000
macro avg	0.80	0.80	0.80	10000
weighted avg	0.80	0.80	0.80	10000

Figure 9: Relative metrics of Quadratic Discriminant Analysis

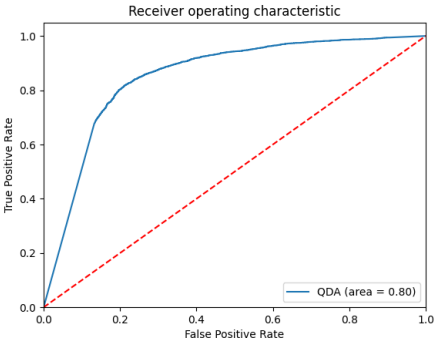

Figure 10: ROC of Quadratic Discriminant Analysis

Hence, LDA performs slightly better than QDA method. For QDA, the precision and recall are smaller than its accuracy, which means there is an imbalance

in the performance of the model in terms of classifying positive and negative sentiments. Therefore, LDA fits our data better.

3.5 K-Nearest Neighbors

In our project, we have opted for the K-Nearest Neighbors (KNN) method due to its simplicity, ease of implementation, and adaptability to various types of datasets. KNN is particularly useful when the underlying data distribution is not well understood or when the relationships between features are complex. In the context of selecting the hyperparameter k, we have chosen a value of approximately $\sqrt{n}$, where n represents the number of data points. This choice aligns with the heuristic of using the square root of the dataset size as it strikes a balance between capturing local patterns in the data and avoiding overfitting. By setting k to $\sqrt{n}$ ($k = 223$), we aim to ensure that the algorithm considers a reasonable number of neighbors for making predictions, avoiding both underfitting and excessive sensitivity to outliers. This approach leverages the benefits of KNN while providing a pragmatic strategy for determining the neighborhood size in our project. Indeed, while K-Nearest Neighbors (KNN) offers simplicity and versatility, it does come with computational challenges, especially when implemented in Python for large datasets. The efficiency concern arises primarily due to the exhaustive search for nearest neighbors, which can be computationally expensive, especially with an increasing number of data points.

For our model, we choose a brute-force search that directly computes the distances between the query point and all points in the dataset, then selects the k nearest neighbors after sorting. The way we calculate distances is by Euclidean distances because this metric is simple to implement and sensitive to the magnitude of differences between each feature.

After applying this model to our dataset, we generated the following results:

	precision	recall	f1-score	support
0	0.75	0.60	0.67	4957
1	0.67	0.81	0.73	5043
accuracy	-	-	0.71	10000
macro avg	0.71	0.71	0.70	10000
weighted avg	0.71	0.71	0.70	10000

Figure 11: ROC of K-nearest neighbors

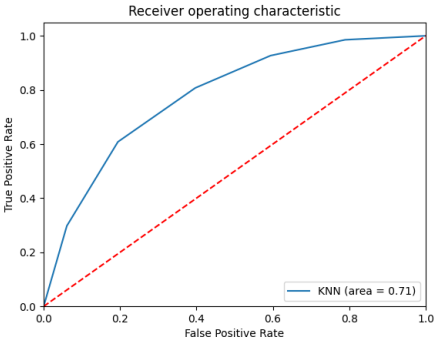

Figure 12: ROC of K-nearest neighbors

Based on the results, we find that KNN model has a higher rate of precision for the positive class and recall for the negative class. This means it performs well in making accurate positive predictions. However, the slightly small f1-score and accuracy represent that this model does not provide a balanced measure. Moreover, KNN is computationally expensive to calculate distances between points in a high-dimensional dataset like ours. Therefore, we may not consider this model.

3.6 Support Vector Machine

We apply the support vector machine to our dataset. This method finds a hyperplane to separate our data linearly into distinct classes with maximal margins (Hsu et al., 2010). We choose to apply the LinearSVC from the scikit learn package. LinearSVC implements a one-vs-rest scheme (Elbagir and Yang, 2018).

Firstly, we select the $l2$ penalty to add a regularization term to penalize large coefficients, preventing overfitting and making the model more robust. Then, we use the squared hinge loss function, a differentiable loss function, to penalize points on the wrong side of the decision boundary (Chu et al., 2002).

After training the Linear Support Vector Classification model, we receive the following results from the testing dataset:

	precision	recall	f1-score	support
0	0.87	0.83	0.85	4957
1	0.84	0.87	0.86	5043
accuracy	-	-	0.85	10000
macro avg	0.85	0.85	0.85	10000
weighted avg	0.85	0.85	0.85	10000

Figure 13: relative metrics of SVM

From this result, we can find that this model has high precision, recall, and accuracy. However, similar to KNN, support vector machines can also be computationally expensive when dealing with large datasets. Therefore, though it performs well, we may still need to do a further comparison with similar methods such as Logistic Regression to generate the final choice.

4 Conclusion and Discussion

4.1 Result Evaluation

We first compare some statistics regarding our models.

	Logistic_Reg	Rand Forest	LDA	QDA	GNB	KNN	SVM
mae	0.1452	0.2291	0.1536	0.2012	0.3207	0.2939	0.1483
mse	0.1452	0.2291	0.1536	0.2012	0.3207	0.2939	0.1483
rmse	0.3811	0.4786	0.3919	0.4486	0.5663	0.5421	0.3851
r2	0.4192	0.0836	0.3856	0.1951	-0.2829	-0.1757	0.4068

Figure 14: Stats Table

Based on these results, we find that the mean square error of Logistic Regression, Linear Discriminant Analysis, and Linear Support Vector Classification are relatively smaller than other models, which means these three models' predictions are closer to the actual data. Also, the R-square of the Quadratic Discriminant Analysis, Random Forest, Gaussian Naive Bayes, and K-Nearest Neighbors are low or negative, showing that these four models may not fit our data.

Besides comparing the statistics, we also create an accuracy table as shown in the following figure.

	Logistic_Reg	Rand Forest	LDA	QDA	GNB	KNN	SVM
accuracy	0.8548	0.7709	0.8464	0.7988	0.6793	0.7061	0.8517

Figure 15: Accuracy Table

Based on this table, the accuracy reflects that Logistic Regression, Linear Discriminant Analysis, and Linear Support Vector Classification have a similarly high accuracy in classifying the reviews into positive and negative groups. Therefore, they are the valid and appropriate model we will use for our dataset.

4.2 Limitation

Firstly, the absence of explicit consideration for potential class imbalance within the sentiment labels. This could potentially introduce bias in the model, favoring the more prevalent sentiment class.

Another limitation lies in the lack of a Cross-Validation strategy. The absence of k-fold cross-validation raises concerns about the reliability of the estimates for the model's performance.

Additionally, the text pre-processing choices of the project, such as lemmatization and special character removal, may impact the models' performance. The decisions during pre-processing could influence the performance of the model.

Finally, the choice of reducing features using Principal Component Analysis (PCA) based on a cumulative explained variance threshold might introduce variability in model performance. The use of this reduction method could potentially affect the results.

4.3 Future Work

1. **Addressing Class Imbalance:** In addition to the future work to solve the limitation of the project, we can address potential class imbalance by employing techniques like oversampling, undersampling, or utilizing specialized algorithms like SMOTE (Synthetic Minority Oversampling Technique). These methods can ensure a more balanced representation of sentiments in the dataset, preventing biases towards the majority class.

2. **Robust Cross-Validation Strategy**: Implementing a robust cross-validation strategy, such as k-fold cross-validation, assumes paramount significance within the experimental framework. This approach will yield more reliable estimates of model performance, promoting confidence in the models' ability to generalize to unseen data.

3. **Handling Pre-processing Challenges**: To tackle challenges stemming from pre-processing choices, particularly lemmatization and special character removal, explore the integration of deep learning models. Architectures like Long Short-Term Memory (LSTM) networks or attention mechanisms are adept at capturing intricate relationships within textual data, offering a more nuanced understanding(Olah, 2015).

4. **Optimizing Feature Reduction:** Fine-tuning the cumulative explained variance threshold in Principal Component Analysis (PCA) is essential. This optimization seeks to strike a balance between effective feature reduction and maintaining optimal model performance.

References

[1] *Bing Liu, Minqing Hu and Junsheng Cheng. "Opinion Observer: Analyzing and Comparing Opinions on the Web." Proceedings of the 14th International World Wide Web conference (WWW-2005), May 10-14, 2005, Chiba, Japan.*

[2] *Chu, W., Ong, C. J., & Keerthi, S. S. (2002). A note on least squares support vector machines. Technical Report, CD-02-09.*

[3] *Elbagir, S., & Yang, J. (2018). Sentiment analysis of Twitter data using machine learning techniques and Scikit-Learn. Proceedings of the 2018 International Conference on Algorithms, Computing and Artificial Intelligence. https://doi.org/10.1145/3302425.3302492*

[4] *Hsu, C. W., Chang, C. C., & Lin, C. J. (2003). A practical guide to support vector classification.*

[5] *Khyani, D., Siddhartha, B. S., Niveditha, N. M., & Divya, B. M. (2021). An interpretation of lemmatization and stemming in natural language processing. Journal of University of Shanghai for Science and Technology, 22(10), 350-357.*

[6] *Martins, C. (2023). Gaussian naive Bayes explained with Scikit-Learn. Built In.*

[7] *Mbaabu, O. (2020). Introduction to random forest in machine learning. Section.*

[8] *Minqing Hu and Bing Liu. "Mining and Summarizing Customer Reviews." Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2004), Aug 22-25, 2004, Seattle, Washington, USA.*

[9] *Olah, C. (2015). Understanding LSTM Networks. colah.github.io/posts/2015-08-Understanding-LSTMs/*

[10] *Pereira, J. M., Basto, M., & Silva, A. F. (2016). The Logistic Lasso and ridge regression in predicting corporate failure. Procedia Economics and Finance, 39, 634-641. https://doi.org/10.1016/s2212-5671(16)3010-0*

[11] *Wold, S., Esbensen, K., & Geladi, P. (1987). Principal component analysis. Chemometrics and Intelligent Laboratory Systems, 2(1-3), 37-52. https://doi.org/10.1016/0169-7439(87)90094-9*