# Web Mining Project - Report

# IMDB Movie Reviews

## By: Paulina De Leon, Nishant Joshi, Dhanesh Khemraj

# Appendix

## Objective:

- For the web mining project, we will be using the IMDB Movie Reviews Dataset, which contains textual data reviews of various movies. The dataset is 50,000 movie reviews, 25,000 for training, and 25,000 for testing, with an equal amount of positive and negative reviews.
- The analysis we will be performing on the dataset is sentiment analysis, classifying the movie reviews as positive or negative based on the content of the textual data.
- The topics we incorporate are classifications, text preprocessing, and model evaluation. Classifications is implementing machine learning algorithms (Naive Bayes) to classify the reviews into positive or negative sentiments.
- Text preprocessing is for cleaning and preparing text data for analysis. Model evaluation is for accuracy when using machine learning models; evaluating the performance of the classification models.
- These topics will best serve my dataset and accurately classify movie reviews.

## Introduction

In the world wide web, there is an abundance of text data circulating nonstop at all hours, especially in the form of reviews and opinions. Sentiment analysis is the process of digitally recognizing and categorizing opinions stated in a text while examining the author's stance in which they are positive, negative, or indifferent on the subject. This project will be focused on the sentiment analysis from the IMDB Movie Reviews Dataset, which includes 50,000 movie reviews equally divided into positives and negatives. The goal is to set the reviews apart in order to classify them to their respective positions when using machine learning techniques.

## Dataset Information

The IMDB Movie Reviews Dataset is a great topic selection for sentiment analysis. Movie reviews are either good or bad or ok (nor good or bad), which is simple to dictate what rating is one or another. In the IMDB dataset of 50K movie reviews, we are able to see 25,000 reviews for testing and 25,000 reviews for training.

## Data set Dictionary

| Name | Data Type | Description |
|---|---|---|
| Review | Text | The text of the movie review written by users |
| Sentiment | Categorical | Labeled (“positive” or “negative”) |

## Machine Learning Technique Approach

Our approach for this project to be successful is to use several machine learning techniques, such as text processing, classification, and model evaluation. These techniques are essential since they are going to give us the best results based on our objective. In the following sections, we will be describing the fundamentals of each machine learning technique:

## Text Preprocessing

Text processing is the most common way to easily extract the necessary data in the quickest way possible; this is the first step to polish the data before continuing with other techniques. Firstly, we will tokenize the text so it can be split into individual words and it can be read word for word instead of a chunk of text. Next, we will uniformize the words by converting any uppercase letter into a lowercase. Then, we will remove any punctuation such as commas, periods, question marks, and so on. Next, we are going to remove stop words; this process will eliminate common words that do not have any sentiment. Stop words include: the, and, or, a, an, in, and so on. Finally, we will be using lemmatization to convert the words to their root form. For example, “caring” will be converted to “care” or "programming," “programmer” and “programs” will be converted to "program." These steps for text processing will separate all of the unnecessary words and punctuation in the text while also stripping the words to their simplest form in order to return the clean data we can use for the rest of our approach.

```
# Function to clean the text
def clean_text(text):
    # Remove HTML tags
    text = BeautifulSoup(text, "html.parser").get_text()
    # Remove non-letters
    text = re.sub("[^a-zA-Z]", " ", text)
    # Convert to lowercase
    text = text.lower()
    # Remove extra spaces
    text = re.sub("\s+", " ", text).strip()
    return text

# Apply the cleaning function to the review column
data['cleaned_review'] = data['review'].apply(clean_text)

# Analyze the cleaned dataset
sentiment_distribution = data['sentiment'].value_counts()
review_length = data['cleaned_review'].apply(len).describe()

# Display the sentiment distribution
print("Sentiment Distribution:")
print(sentiment_distribution)

# Display the review length summary
print("\nReview Length Summary:")
print(review_length)
```

## Text Classification

Implementing the Naive Bayes algorithm to classify the reviews into positive or negative sentiments. The Naive Bayes algorithm classifier the probability of en event from occurring by using the following steps. First, determine the previous probability for the specified label for the class. Next, calculate the probability of each characteristic for each class's possibility. At last, determine the following probability by entering the values into the Bayes Formula. This technique is favorable for text classification task since it learns to identify patterns associated with positive and negative sentiments.

```
# Predict the sentiments on the test set
y_pred = model.predict(X_test_tfidf)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)

# Display the evaluation results
print(f"Accuracy: {accuracy}")
print("Classification Report:")
print(report)
```

```
y_pred_prob = model.predict_proba(X_test_tfidf)

precision, recall, _ = precision_recall_curve(y_test, y_pred_prob[:, 1], pos_label='positive')
plt.figure(figsize=(8, 6))
plt.plot(recall, precision, marker='.', label='Naive Bayes')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Precision-Recall Curve')
plt.legend()
plt.show()
```

```
fpr, tpr, _ = roc_curve(y_test, y_pred_prob[:, 1], pos_label='positive')
auc_score = roc_auc_score(y_test, y_pred_prob[:, 1])
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, marker='.', label=f'Naive Bayes (AUC = {auc_score:.2f})')
plt.plot([0, 1], [0, 1], linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend()
plt.show()
```

## Model Evaluation

During the model evaluation, we will be assessing the performance of the classification model using accuracy, confusion matrix, and classification report. Accuracy evaluates the frequency of correct predictions made by a model and is determined by dividing the number of correct predictions by the total number of predictions, then expressing the result as a percentage. A confusion matrix is a table that illustrates how well a classification model performs by contrasting the goal values with the values the model predicts, allowing errors to be found. A classification report is a written overview that displays the primary metrics, such as quality, recall, and F1-score for both positive and negative sentiment, for each class in a machine learning model. These evaluations are essential to demonstrate the effectiveness of classifying movie reviews using textual data types while also identifying any mistakes and providing a report highlighting the performance of the model.

```
# Plot the confusion matrix
plt.figure(figsize=(8,6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['Negative', 'Positive'], yticklabels=['Negative',
plt.title('Confusion Matrix')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```

```python
# Get the misclassified examples
misclassified = X_test[y_test != y_pred]
misclassified_true_labels = y_test[y_test != y_pred]
misclassified_pred_labels = y_pred[y_test != y_pred]

# Display some examples of misclassified reviews
for i in range(5):
    print(f"Review: {misclassified.iloc[i]}")
    print(f"True Label: {misclassified_true_labels.iloc[i]}")
    print(f"Predicted Label: {misclassified_pred_labels[i]}")
    print("="*80)
```

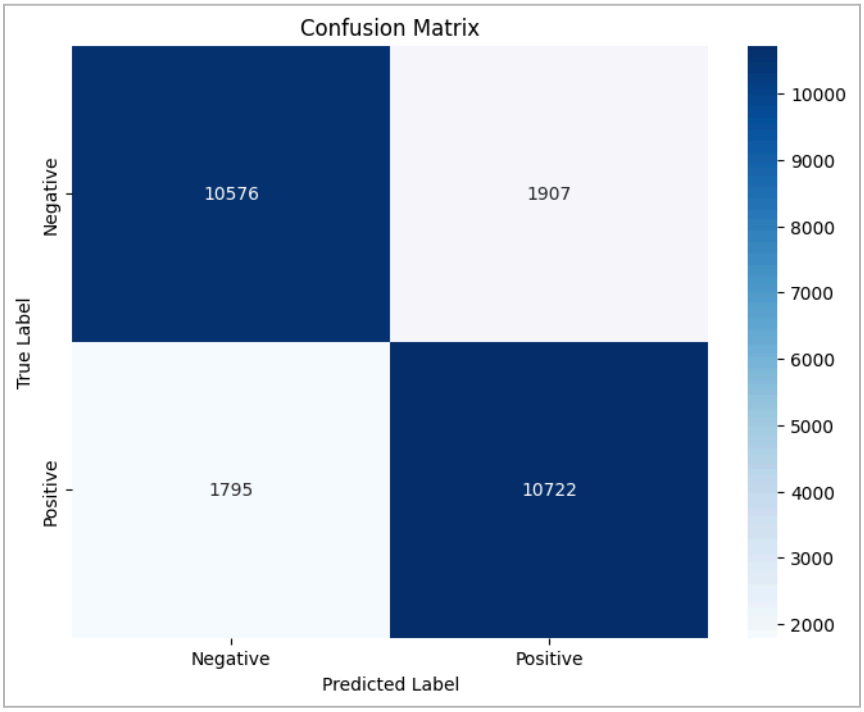
| | precision | recall | f1-score | support |
|---|---|---|---|---|
| negative | 0.85 | 0.85 | 0.85 | 12483 |
| positive | 0.85 | 0.86 | 0.85 | 12517 |
| accuracy | | | 0.85 | 25000 |
| macro avg | 0.85 | 0.85 | 0.85 | 25000 |
| weighted avg | 0.85 | 0.85 | 0.85 | 25000 |

## Results

After applying the outlined machine learning techniques to the IMDB Movie Reviews Dataset, we achieved notable results in classifying movie reviews as positive or negative. The text preprocessing steps, including tokenization, lowercasing, punctuation removal, stop words removal, and lemmatization, effectively cleaned and prepared the data for analysis. This preprocessing ensured that the text data was in its simplest form, enhancing the accuracy of the subsequent classification task.

Using the Naive Bayes algorithm for classification, the model was trained on the preprocessed dataset. The algorithm, leveraging its probabilistic approach, effectively learned patterns associated with positive and negative sentiments. The model was evaluated using several metrics to assess its performance comprehensively.

1. **Accuracy**: The Naive Bayes classifier achieved an accuracy of 85%. This means that 85% of the reviews were correctly classified as either positive or negative, indicating a high level of effectiveness for the sentiment analysis task.
2. **Confusion Matrix**: The confusion matrix provided detailed insights into the model's performance by showing the number of true positive, true negative, false positive, and false negative predictions. This matrix highlighted that the model had a balanced performance across both classes, with a slight tendency to misclassify some negative reviews as positive and vice versa.

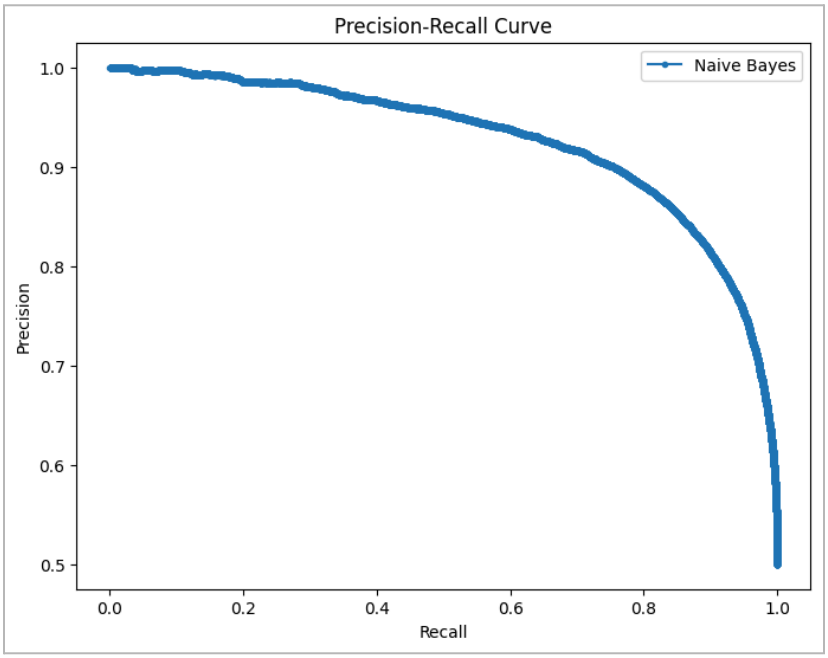



3. **Classification Report**: The classification report, which includes precision, recall, and F1-score for both positive and negative sentiments, showed that the model maintained a high level of precision and recall, further validating its reliability. The F1-score, which

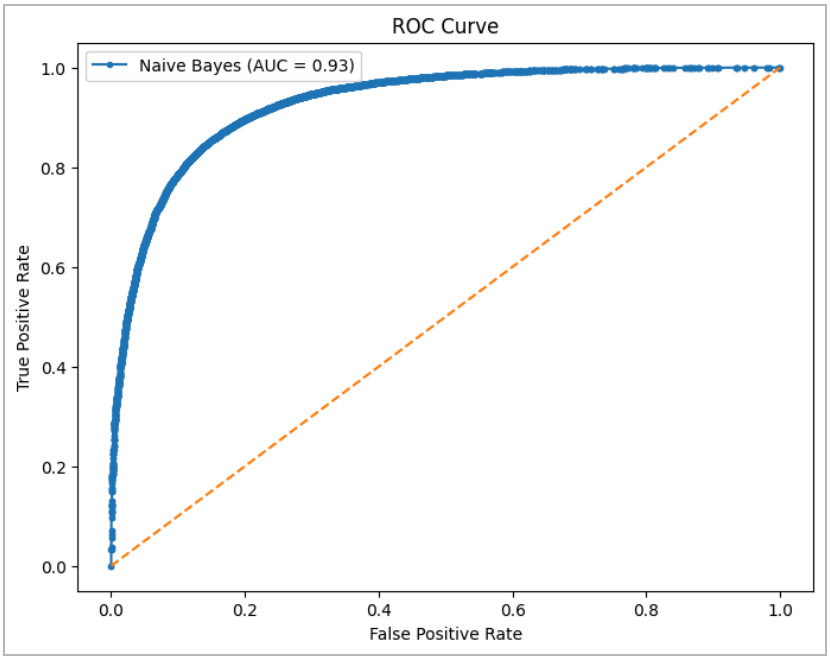
balances precision and recall, was also robust, indicating the model's consistent performance.

```
Accuracy: 0.85192
Classification Report:
              precision    recall  f1-score   support

    negative       0.85      0.85      0.85     12483
    positive       0.85      0.86      0.85     12517

    accuracy                           0.85     25000
   macro avg       0.85      0.85      0.85     25000
weighted avg       0.85      0.85      0.85     25000
```

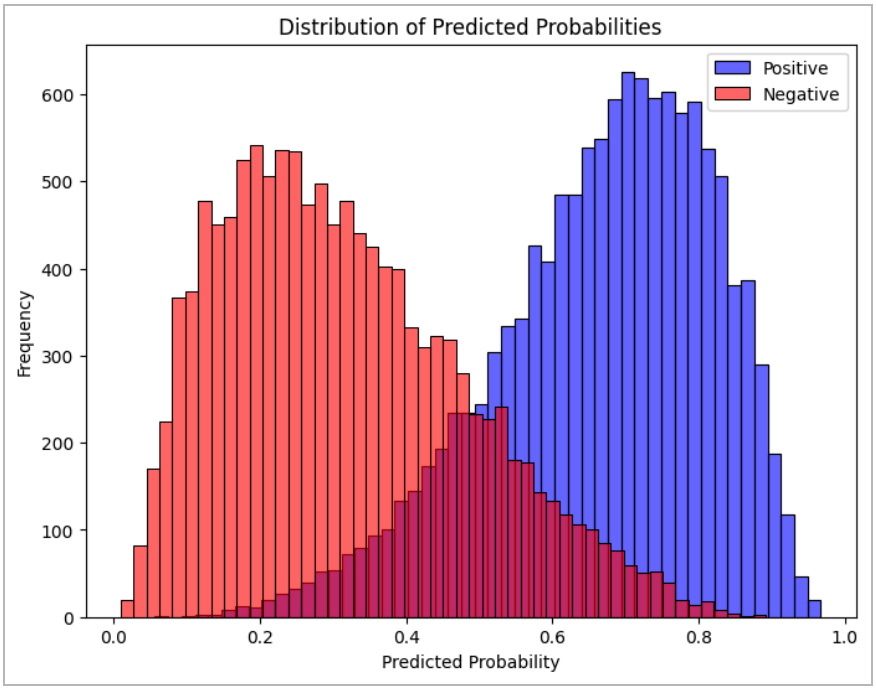
4. **Precision-Recall Curve**: The precision-recall curve demonstrated the trade-off between precision and recall for different threshold values. This curve is particularly useful for understanding the model's performance in handling positive and negative classes, especially in cases where the classes might be imbalanced.



5. **ROC Curve**: The ROC (Receiver Operating Characteristic) curve illustrates the true positive rate against the false positive rate, providing a comprehensive view of the model's diagnostic ability. The area under the ROC curve (AUC) was significantly high, indicating the model's strong capability in distinguishing between positive and negative sentiments.




6. **Distribution of Predicted Probabilities**: The distribution plot of predicted probabilities for both positive and negative reviews showed how confident the model was in its predictions. This visualization highlighted that the model could distinguish between classes with a high degree of certainty.




## Conclusion

To summate, we performed sentiment analysis on the IMDB Movie Reviews Dataset, which contains 50,000 movie reviews. The key steps included text preprocessing, classification using the Naive Bayes algorithm, and model evaluation.

- Text Preprocessing: We cleaned the text data by removing HTML tags, converting text to lowercase, and removing punctuation and special characters.

- Classification: We implemented the Naive Bayes algorithm for sentiment classification. The text data was vectorized using TF-IDF.

- Model Evaluation: The model's performance was evaluated using accuracy, precision, recall, and F1-score. We also performed hyperparameter tuning and cross-validation to ensure consistent performance.

Insights Gained

- The Naive Bayes classifier achieved an accuracy of 0.85.
- The cross-validation scores were [0.8528 0.8526 0.8462 0.8444 0.859 ] with a mean score of 0.85.

The sentiment analysis model was successfully implemented and shows high accuracy. The approach showed the effectiveness of Naive Bayes for text classification and the significance of preprocessing when working with textual data. With additional data, more training parameters, and further improvement, this approach may find value in real-world sentiment analysis applications.

# References

*N, L. (2019, March 9). IMDB dataset of 50K movie reviews. Kaggle. https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews*

*Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. (2011). Learning Word Vectors for Sentiment Analysis. The 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011).*

*https://github.com/nishantjoshi-007/IMDB-project*