Lab Report 1: Classification by Naïve Bayes Classifier and SVM

Name: Ruotong Yang

Group Number: 6

Group Members: Ruotong Yang, Runze Cui, Yilai Chen

September 19, 2025

1 Introduction

The purpose of this lab is to build a sentiment classification system for short text reviews, predicting whether a review is positive or negative. The work is divided into two parts: first, an exploratory data analysis (EDA) was performed to understand the dataset, examine class balance, review lengths, and vocabulary distributions, and identify challenges such as stopwords, HTML artifacts, and noisy tokens. These insights guided the design of the preprocessing pipeline. In the second part, we developed and compared different models, including Naïve Bayes, Support Vector Machine (SVM), and a transformer-based model (DistilBERT), to evaluate the trade-offs between classical machine learning and modern deep learning. The models were trained on the provided dataset, validated on a held-out split, and finally tested on an external Yelp dataset to assess their generalization capability.

2 Exploratory Data Analysis (EDA)

2.1 Dataset Overview

The dataset consists of three parts: 7,500 training samples, 2,500 test samples, and 5,000 evaluation samples. Each sample contains two columns: `score` (integer, the sentiment label) and `text` (string, the review content). No missing values were found, and the dataset is clean and ready for analysis.

2.2 Target Variable (score)

The target variable is binary, with 3,752 positive reviews and 3,748 negative reviews in the training set. The distribution is nearly perfectly balanced, which helps to avoid bias during model training.

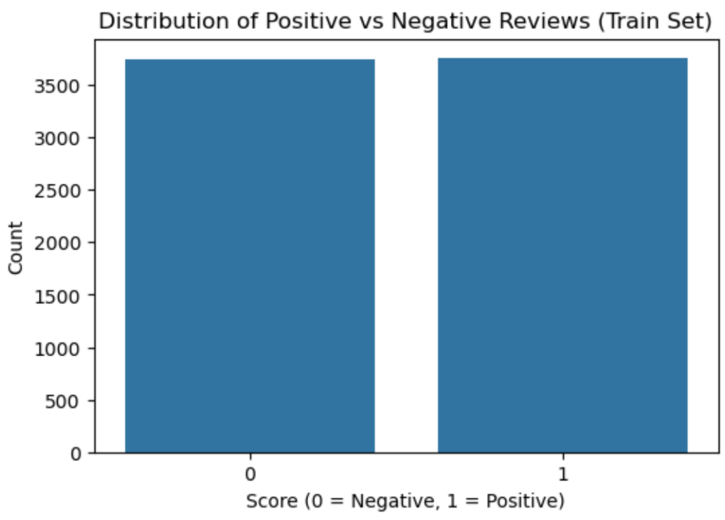

Figure 1: Distribution of positive and negative reviews in the training set.

2.3 Review Length Analysis

The average review length is 595.55 characters (median = 234, min = 11, max = 7,126). In terms of words, the average review contains 105.31 words (median = 43, min = 1, max = 1,164). Most reviews are relatively short (10–40 words), while a few very long reviews act as outliers.

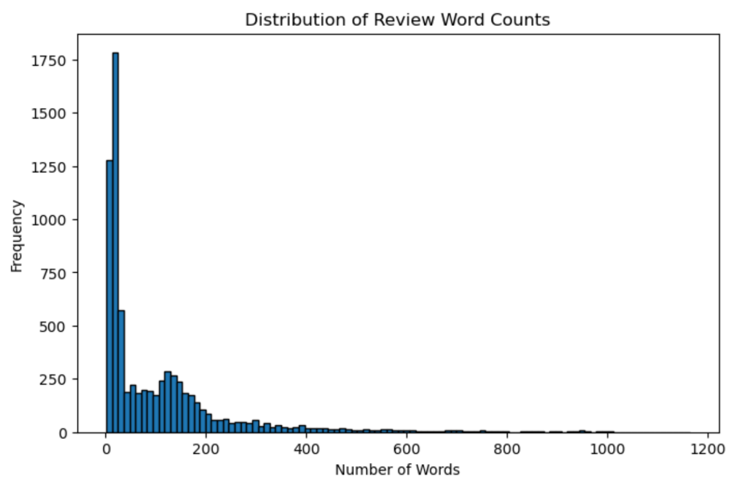

Figure 2: Distribution of positive and negative reviews in the training set.

2.4 Word Frequency Analysis

The most common tokens include many stopwords such as *the*, *and*, *to*, as well as artifacts like *br* from HTML tags. Contraction fragments (e.g., *s*, *t*) also appear due to tokenization. These observations indicate that preprocessing is necessary to remove noise and highlight meaningful words.

2.5 Positive vs. Negative Vocabulary

Positive reviews frequently contain words such as *great*, *good*, *best*, and *love*, while negative reviews contain sentiment-bearing terms such as *bad* and *worst*. Both positive and negative reviews share domain words like *movie*, *film*, and *story*. This contrast shows that sentiment-indicative vocabulary provides strong signals for classification.

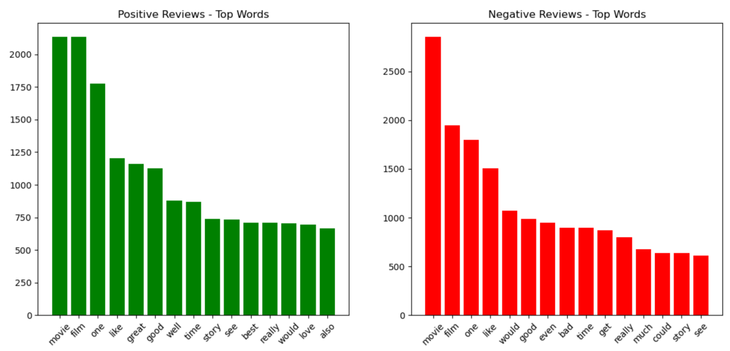

Figure 3: Distribution of positive and negative reviews in the training set.

2.6 Noise and Preprocessing Needs

Examples of noisy text include HTML tags (`<br />`), mentions (e.g., `@user`), hashtags (e.g., `#topic`), and emojis. Based on these findings, the following preprocessing steps are proposed:

- Lowercase all text.
- Remove HTML tags (e.g., `<br>`).
- Remove mentions (@username) and hashtags (#topic).

- Remove punctuation, numbers, and extra spaces.
- Remove stopwords (e.g., *the*, *and*, *to*).
- Optionally apply stemming or lemmatization.

2.7 Reflection

The main challenges of the dataset include the large variation in review length, the dominance of stopwords, and the presence of noisy artifacts such as HTML tags and contraction fragments. These findings directly inform the preprocessing stage: consistent lowercasing, noise removal, and stopword filtering will be essential. After cleaning, feature extraction with Bag-of-Words or TF-IDF will better capture sentiment-bearing words, improving the effectiveness of classifiers such as Naïve Bayes and SVM.

3 Data Cleaning & Feature Extraction

3.1 Data Cleaning

Based on our EDA findings, we applied a comprehensive cleaning pipeline to the raw text. The required steps are listed below, followed by examples in Table 1.

- **Lowercasing:** All text was converted to lowercase.
- **Noise Removal:** We removed HTML tags (e.g., `<br>`), mentions (@), hashtags (#), punctuation, and numbers.
- **Stopword Removal:** Common English stopwords (e.g., "the", "a", "is") were filtered out.
- **Lemmatization:** We applied lemmatization to reduce words to their dictionary root form (e.g., "applying" to "apply"), creating a cleaner vocabulary.

Table 1: Examples of Raw vs. Cleaned Text.

Raw Text (Snippet)	**Cleaned Text (Snippet)**
overgeneralized, not helpful to anyone seriously applying: if you as a premed know nothing, i mean nothing about the medical school admissions process, then this book can be used for a start. i found ...	overgeneralize helpful anyone seriously apply premed know nothing mean nothing medical school admission process book use start find...
Great sound and service...	great sound service...
love this book!!!: this book is a fast read about a poor young farmer boy who grew up to be an american hero, withstanding many trials along the way...also an insite into what our ww 00 veterans went ...	love book book fast read poor young farmer boy grow american hero withstand many trial along way also insite ww veteran go...

3.2 Feature Extraction

We converted the cleaned text into numerical features using the **TF-IDF** method.

Justification: We chose TF-IDF over a simple Bag-of-Words because it effectively highlights words that are important for sentiment by giving them higher scores, while down-weighting common, less informative words.

Feature Selection: To create a focused and robust feature set, we limited the vocabulary to the **top 5,000 features** based on their TF-IDF scores. We also included bi-grams (`ngram_range=(1,2)`) to capture phrases like "not good". This resulted in a vocabulary size of 5,000.

4 Model Development

4.1 Models Used

For this classification task, we developed and compared two different models:

- **Naïve Bayes Classifier (NBC):** We used the `MultinomialNB` implementation from scikit-learn. This probabilistic model is highly efficient and performs well on text classification tasks, especially with features like TF-IDF counts.
- **Support Vector Machine (SVM):** We used `LinearSVC` from scikit-learn as our baseline SVM model, which is well-suited for high-dimensional sparse text data. The model was trained on the TF-IDF features using default parameters (`C=1.0`, `loss='squared_hinge'`).
- **BERT (DistilBERT):** As an advanced baseline, we fine-tuned a pre-trained distilbert-base-uncased model from HuggingFace Transformers for binary sentiment classification.

4.2 Training and Validation Process

The models were trained exclusively on the provided training dataset (7,500 samples). The separate validation dataset (2,500 samples) was used to evaluate the performance of each model on unseen data. This process allows for a fair and unbiased comparison between the Naïve Bayes classifier, the Support Vector Machine and BERT. For SVM, the validation set was also used to tune its hyperparameters.

4.3 Hyperparameter Tuning

The Naïve Bayes model was trained using its default hyperparameters, including a Laplace smoothing parameter of `alpha=1.0`, which is a robust standard.

For the SVM model, we performed a simple grid search over four values of the regularization parameter $C \in \{0.01, 0.1, 1.0, 10.0\}$. We trained a separate model for each C and selected the one with the highest F1-score on the validation set. The best-performing value was then used to retrain the final SVM model on the training data.

For the BERT model (DistilBERT), we fine-tuned the pre-trained transformer on our training set with the following hyperparameters: a learning rate of 5×10^{-5}, batch size of 16 for training and 32 for evaluation, weight decay of 0.01, and 2 training epochs (initially for quick experimentation, with the option to increase to 3–4 epochs for improved performance).

5 Evaluation

5.1 Evaluation Metrics

We evaluated our models using two primary metrics: Accuracy and F1-Score.

Accuracy was chosen because it provides an intuitive measure of the overall correct predictions, which is reliable for our balanced dataset.

F1-Score was chosen as it provides a robust balance between Precision (the model's exactness) and Recall (the model's completeness), making it an excellent measure of a model's overall effectiveness on both classes.

5.2 Performance Results

Table 2 summarizes the performance of both models on the validation and the external Yelp evaluation datasets.

Table 2: Model Performance on Validation and Evaluation Datasets.

Dataset	**Model**	**Accuracy**	**F1-Score**
Validation Set	Naïve Bayes	83.72%	83.67%
Validation Set	SVM (C=1.0)	84.28%	83.97%
Validation Set	BERT (DistilBERT)	86.96%	86.91%
Yelp Evaluation Set	Naïve Bayes	82.76%	82.54%
Yelp Evaluation Set	SVM (C=1.0)	82.98%	83.34%
Yelp Evaluation Set	BERT (DistilBERT)	86.00%	86.57%

5.3 Visualization

The confusion matrices provide a detailed breakdown of classification performance. Figure 4 shows the model's performance on the validation set, while Figure 5 shows its performance on the out-of-domain Yelp evaluation set.

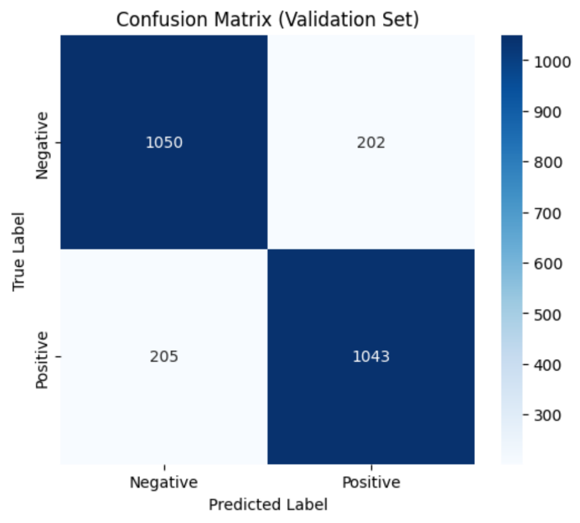

Figure 4: Confusion Matrix on the Validation Set.

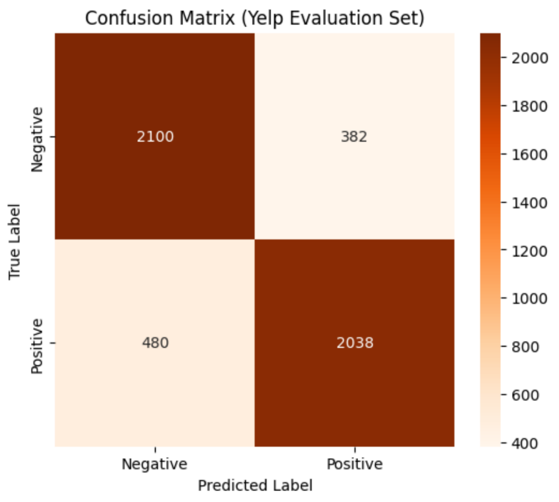

Figure 5: Confusion Matrix on the Yelp Evaluation Set.

5.4 Discussion

The Naïve Bayes classifier achieved a strong baseline performance with an accuracy of 83.72% on the validation set. When tested on the out-of-domain Yelp review dataset, its performance showed only a minor decrease to 82.76%, indicating good generalization capabilities. The model effectively learned sentiment features that are applicable beyond the original training data's domain.

Compared to Naïve Bayes, the Support Vector Machine (SVM) with $C = 1.0$ achieved slightly higher validation accuracy (84.28%) and F1-score (83.97%), showing that margin-based classifiers can better separate positive and negative reviews when using TF-IDF features. However, its performance on the Yelp dataset (82.98% accuracy, 83.34% F1) suggests that the improvement is modest and does not dramatically reduce the impact of domain shift. This indicates that SVM, while more expressive, still relies heavily on lexical patterns present in the training data.

BERT (DistilBERT), on the other hand, delivered a significant boost in performance, achieving 86.96% accuracy on the validation set and 86.00% on the Yelp dataset. The smaller generalization gap shows that contextual embeddings allow BERT to capture more robust sentiment cues, including word order and nuanced phrases like negation, which TF-IDF-based models may miss. This makes BERT the most effective model in our experiments, though it comes with higher computational cost during training and inference.

6 Reflection & Conclusion

In this lab, we compared Naïve Bayes, SVM, and BERT models for sentiment classification. Naïve Bayes provided a solid baseline, SVM achieved a slight improvement by optimizing the margin, and BERT delivered the highest accuracy and F1-score, especially on the out-of-domain Yelp dataset, showing superior generalization.

However, BERT required significantly more computation and training time compared to the classical models, and our SVM tuning was limited to a simple grid search. With more training epochs or advanced hyperparameter optimization, performance could be further improved.

This lab helped us understand the trade-off between classical machine learning and modern deep learning models, as well as the importance of preprocessing, feature extraction, and proper model tuning for achieving strong results.

7 References

References

[1] ChatGPT, OpenAI. (2025). *Assistance with debugging, model implementation, and Latex formatting.*

[2] Cortes, C., Vapnik, V. (1995). *Support-vector networks.* Machine Learning, 20(3), 273–297. https://doi.org/10.1007/BF00994018

[3] Ramos, J. (2003). *Using TF-IDF to determine word relevance in document queries.* Proceedings of the First Instructional Conference on Machine Learning.

[4] Devlin, J., Chang, M. W., Lee, K., Toutanova, K. (2019). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.* NAACL-HLT. https://arxiv.org/abs/1810.04805

[5] Sanh, V., Debut, L., Chaumond, J., Wolf, T. (2019). *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter.* arXiv preprint arXiv:1910.01108.