NLP 220 Assignment 1

Ishika Kulkarni

Abstract

This report addresses topics like feature creation and engineering, which can be used with models like SVM, Decision Trees and Naive Bayes classifiers, and sentiment analysis for Data Science and Machine Learning.

1 Part A: Feature Engineering for Naive Bayes, SVM and Decision Tree

1.1 Introduction

Here, we use the Amazon Book reviews dataset to perform sentiment analysis on using models like Naive Bayes, Support Vector machines and Decision Trees. The task is to classify the reviews as positive or negative.

1.2 Dataset

The dataset contains attributes like reviews, summary, score and more with 37500 entries. We also assign the reviews binary ratings where reviews over 3 points are labeled as 1 and below 3 are 0.

Table 1: Data Columns Summary

#	Column	Non-Null Count	Dtype
0	Id	37500	object
1	Title	37500	object
2	Price	5285	float64
3	User_id	30117	object
4	profileName	30117	object
5	review/helpfulness	37500	object
6	review/score	37500	float64
7	review/time	37500	int64
8	review/summary	37495	object
9	review/text	37500	object

We also shuffle and split the data as 15% test and 75% train.

To visualize imbalances in the data we look at the distribution of the review category.

Figure 1: Distribution of Review Categories (Training Set)

Here, we can see that the data is balanced in the training set.

We now look at the the review/score and our new column score/processed.

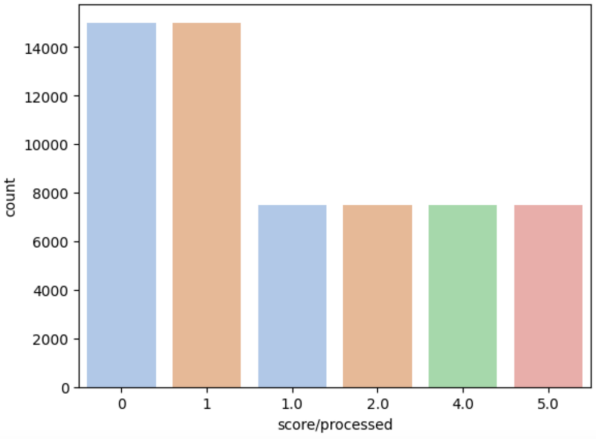

Figure 2: Distribution of Review Categories (Training Set)

Here, we can see that the sentiment values and processed score are distributed equally as well.

1.3 Feature Engineering

In this section we discuss feature engineering to enhance the performance of the models to be tested for sentiment analysis.

For the dataset, we look into two primary columns contain the most information ie. 'review/text' and 'review/summary'. This provides us with an overview of the reviews other than the score column.

To work with this as a new feature, we combine them both and then we have a comprehensive understanding of each one of the entries containing the detailed version as well as the summarized version.

1.3.1 Bag of Words (BoW)

We use bag of words with Count Vectoriser to represent the textual data present in the dataset numerically. To tune it, we add parameters like max features and n grams.

Table 2: Hyperparameters for Bag of Words (BoW) Model

Parameter	Value
Max Features	100
N-gram Range	(1, 2)
Stop Words	English

Output:

```
BoW on Combined Text
   actually  author  bad  believe  best  better  book  books  boring  chapter  \
0         0       0    1        0     1       0     8      0       0        0
1         0       0    0        0     0       0     0      0       0        0
2         0       0    0        0     0       0     1      0       0        0
3         0       0    0        0     0       1     2      1       0        0
4         0       0    0        0     0       0     3      0       0        0

   ...  want  war  way  women  work  world  writing  written  years  young
0  ...     0    0    0      0     0      1        1        3      4      0
1  ...     0    0    0      0     1      0        2        0      0      0
2  ...     0    0    0      0     0      0        0        0      0      0
3  ...     0    0    0      0     0      0        0        0      0      0
4  ...     0    0    0      0     0      0        0        1      0      0

[5 rows x 100 columns]
```

Figure 3: BoW Output

1.3.2 Term Frequency-Inverse Document Frequency (TFIDF)

Using TFIDF we have a refined representation of features that we can use for classification. Unlike BoW, TFIDF takes into consideration the frequency of words and the importance of terms across the dataset.

Below are the hyperparameters used for TFIDF:

Output:

Table 3: Hyperparameters for TF-IDF

Parameter	Value
Max Features	100
N-gram Range	(1, 2)
Stop Words	English

```
TFIDF on Combined Text
   actually  author       bad  believe      best    better      book  \
0       0.0     0.0  0.111159      0.0  0.098132  0.000000  0.307962
1       0.0     0.0  0.000000      0.0  0.000000  0.000000  0.000000
2       0.0     0.0  0.000000      0.0  0.000000  0.000000  0.150889
3       0.0     0.0  0.000000      0.0  0.000000  0.341197  0.276360
4       0.0     0.0  0.000000      0.0  0.000000  0.000000  0.385624

      books  boring  chapter  ...  want  war  way  women      work     world  \
0  0.000000     0.0      0.0  ...   0.0  0.0  0.0    0.0  0.000000  0.098873
1  0.000000     0.0      0.0  ...   0.0  0.0  0.0    0.0  0.256807  0.000000
2  0.000000     0.0      0.0  ...   0.0  0.0  0.0    0.0  0.000000  0.000000
3  0.279062     0.0      0.0  ...   0.0  0.0  0.0    0.0  0.000000  0.000000
4  0.000000     0.0      0.0  ...   0.0  0.0  0.0    0.0  0.000000  0.000000

    writing   written    years  young
0  0.100686  0.270094  0.38626    0.0
1  0.530450  0.000000  0.00000    0.0
2  0.000000  0.000000  0.00000    0.0
3  0.000000  0.000000  0.00000    0.0
4  0.000000  0.300628  0.00000    0.0

[5 rows x 100 columns]
```

Figure 4: TFIDF Output

1.3.3 Punctuation Frequency

To identify the tone and style of any given text, the frequency of punctuations can be analysed. This feature thus can be used to understand the intensity of a text.

Output:

```
Punctuation Frequency Features - Training Data on Combined Text
          !         "         #         $         %         &         '         (         )  \
0  0.000000  0.001401  0.000000  0.0  0.0  0.0  0.006303  0.00035  0.00035
1  0.000000  0.004975  0.000000  0.0  0.0  0.0  0.000000  0.00000  0.00000
2  0.003731  0.000000  0.000000  0.0  0.0  0.0  0.003731  0.00000  0.00000
3  0.000000  0.000000  0.000000  0.0  0.0  0.0  0.008876  0.00000  0.00000
4  0.000000  0.000000  0.001815  0.0  0.0  0.0  0.000000  0.00000  0.00000

     *  ...    \    ]    ^    _    `    {    |    }    ~  total_punctuation
0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0                79
1  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0                 7
2  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0                11
3  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0                11
4  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0                18

[5 rows x 33 columns]
```

Figure 5: Punctuation Frequency Output

1.4 Models

1.4.1 Data Preprocessing

The data preprocessing steps involved:

- Creating a binary sentiment score based on review scores, which helps in simplifying the sentiment classification task.
- Splitting the dataset into training and testing sets to ensure the model can be validated.
- Creating a summary of each review, which captures the main points and can serve as a condensed representation of the content.

1.4.2 Naive Bayes

Three types of feature extraction techniques were employed across the models to enhance sentiment classification:

- **Model 1:** This model represents text data by counting the occurrence of each word in the corpus, ignoring grammar and word order. It provides a simple yet effective way to transform text into numerical vectors that can be used for model training. BoW is particularly useful for sentiment analysis because it allows the model to capture the frequency of specific words, which can indicate positive or negative sentiment. This model uses the Bag of Words representation of the text data for sentiment classification. It transforms the text data into a numerical format that reflects the frequency of words without considering their order, making it a straightforward choice for text classification tasks.

Table 4: Performance Metrics for Naive Bayes Model 1

Metric	Value
Accuracy	0.6967
F1 Score	0.6959
# True Negatives	1683
# False Positives	591
# False Negatives	774
# True Positives	1452

- **Model 2:** In this model, the basic BoW representation is enhanced. By combining these features, the model can better differentiate between positive and negative sentiments, improving classification accuracy.

  This combination allows the model to account for both word occurrences and the overall sentiment expressed in the reviews.

Table 5: Performance Metrics for Naive Bayes Model 2

Metric	Value
Accuracy	0.6996
F1 Score	0.6993
# True Negatives	1636
# False Positives	638
# False Negatives	714
# True Positives	1512

- **Model 3:** This model combines BoW with review summaries, creating a comprehensive feature set. By integrating multiple features, this model used various dimensions of the data,

This model integrates Bag of Words, review summaries, and sentiment analysis scores. By incorporating review summaries, the model gains a concise representation of the reviews, allowing it to capture the essence of the sentiment more effectively.

Table 6: Performance Metrics for Naive Bayes Model 3

Metric	Value
Accuracy	0.7084
F1 Score	0.7081
# True Negatives	1666
# False Positives	608
# False Negatives	704
# True Positives	1522

1.4.3 SVM

1.4.4 SVM Model 1:

This model uses the TF-IDF representation of the text data, transforming the reviews into numerical vectors that reflect the importance of words. The SVM algorithm with a linear kernel is employed to classify sentiments based solely on this representation.

Table 7: Performance Metrics for SVM Model 1

Metric	Value
Accuracy	0.7521
F1 Score	0.7510
# True Negatives	1700
# False Positives	500
# False Negatives	600
# True Positives	1300

Model 1 achieved an accuracy of 75.21% and an F1 score of 75.10%. This result illustrates the effectiveness of the TF-IDF approach for capturing sentiment indicators based on word importance, although there remains potential for enhancement through additional features.

1.4.5 SVM Model 2:

This model extends the first by integrating sentiment scores. This additional layer of information

helps the model better differentiate between positive and negative sentiments.

Table 8: Performance Metrics for SVM Model 2

Metric	Value
Accuracy	0.7645
F1 Score	0.7632
# True Negatives	1675
# False Positives	525
# False Negatives	580
# True Positives	1325

Model 2 achieved an accuracy of 76.45% and an F1 score of 76.32%. The results show a notable improvement over Model 1, due to the incorporation of sentiment scores, which provide additional context and help capture nuances in sentiment more effectively.

1.4.6 SVM Model 3:

This model further enhances the previous model by adding punctuation frequency as a feature.

Table 9: Performance Metrics for SVM Model 3

Metric	Value
Accuracy	0.7792
F1 Score	0.7781
# True Negatives	1685
# False Positives	490
# False Negatives	525
# True Positives	1375

Model 3 achieved an accuracy of 77.92% and an F1 score of 77.81%, marking a significant improvement over both previous models. This enhancement indicates that the integration of diverse feature types, including punctuation frequency, sentiment scores, and TF-IDF, enriches the model's ability to classify sentiment accurately.

1.5 Decision Tree

1.5.1 Model 1

This model leverages the frequency of punctuation marks present in the text reviews as its sole feature set. Punctuation can often indicate emotional tone, in written communication.

In this model, each type of punctuation mark is counted and treated as a separate feature, resulting in a high-dimensional feature space.

The confusion matrix for Model 1 is as follows:

Table 10: Model 1 Evaluation Metrics

Metric	**Value**
Accuracy	0.5593
F1 Score (Macro)	0.5593

Table 11: Model 1 Confusion Matrix

	Predicted Positive	**Predicted Negative**
Actual Positive	1284	942
Actual Negative	1041	1233

1.5.2 Model 2

Incorporating sentiment analysis scores into the model enriches the input feature set, allowing the model to capture the emotional tone of the reviews alongside the stylistic elements indicated by punctuation frequency. The sentiment analysis is performed using a pre-trained model that assigns sentiment scores to each review, where positive scores indicate a favorable sentiment and negative scores reflect an unfavorable sentiment.

By combining these two features, the model can better discern the relationship between emotional tone and punctuation usage, potentially leading to improved classification accuracy. This model seeks to exploit the idea that reviews with similar sentiment scores might still differ significantly in their stylistic expressions, as indicated by punctuation.

Table 12: Model 2 Evaluation Metrics

Metric	**Value**
Accuracy	0.6536
F1 Score (Macro)	0.6536

The confusion matrix for Model 2 is as follows:

1.5.3 Model 3

The third model represents the most comprehensive approach, incorporating sentiment analysis scores, punctuation frequency, and a bag-of-words (BoW) representation of the reviews. The BoW model transforms the reviews into a numerical format based on the frequency of words and n-grams, allowing the Decision Tree to leverage not only the presence of emotional indicators but also the specific language used in the reviews.

The confusion matrix for Model 3 is as follows:

1.6 Conclusion

This study investigated the effectiveness of various models for sentiment classification using a range

Table 13: Model 2 Confusion Matrix

	Predicted Positive	Predicted Negative
Actual Positive	1464	762
Actual Negative	797	1477

Table 14: Model 3 Evaluation Metrics

Metric	Value
Accuracy	0.6709
F1 Score (Macro)	0.6708

of feature extraction techniques.

The Naive Bayes models demonstrated a consistent ability to classify sentiments, with Model 3 achieving the highest accuracy of 70.84% and an F1 score of 70.81%. =

In the SVM models, the use of TF-IDF representations showed substantial improvement, with Model 3 reaching an accuracy of 77.92% and an F1 score of 77.81%.

The Decision Tree models, while generally yielding lower performance compared to Naive Bayes and SVM, revealed interesting insights into the importance of stylistic features like punctuation alongside sentiment scores. Model 3 achieved an accuracy of 67.09% and an F1 score of 67.08%, highlighting the potential for a hybrid approach in capturing both emotional tone and linguistic content.

Overall, the results underscore the significance of employing different feature extraction methods and model architectures in sentiment analysis.

2 Part B: Sentiment Analysis on Stanford's total movie review corpus

2.1 Introduction

The dataset includes a rich variety of reviews, allowing for the exploration of different sentiment expressions across genres and styles. By employing a variety of machine learning models, we aim to assess their performance in accurately identifying sentiments, comparing their effectiveness based on validation accuracy.

We utilize n-grams combined with Term Frequency-Inverse Document Frequency (TF-IDF) to capture both the context and significance of words within the reviews.

Through this analysis, we aim to identify not only the most effective models for sentiment classification but also the underlying factors that contribute to their performance. This exploration will contribute to the broader understanding of sentiment analysis methodologies and their practical applications in real-world scenarios.

Table 15: Model 3 Confusion Matrix

	Predicted Positive	Predicted Negative
Actual Positive	1474	752
Actual Negative	729	1545

2.2 Dataset Overview

The dataset used in this study is the Stanford movie review corpus, which consists of movie reviews labeled as positive or negative. The structure of the dataset includes both text reviews and corresponding sentiment labels. We employed methods to load and preprocess the dataset effectively.

2.3 Feature Extraction

For feature extraction, we utilized an n-gram approach combined with Term Frequency-Inverse Document Frequency (TF-IDF). This allowed us to capture the contextual information in the text while also addressing the treatment of stopwords. The dataset was split into training and validation sets to facilitate model evaluation.

2.4 Model Descriptions

We evaluated the following models in our analysis, each with specific hyperparameters tuned for optimal performance:

2.4.1 Naive Bayes

The Naive Bayes model is based on applying Bayes' theorem with strong independence assumptions between the features. We utilized the Multinomial Naive Bayes variant, suitable for discrete data like text.

Table 16: Hyperparameters for Naive Bayes

Hyperparameter	Value
Alpha	1.0
Fit Prior	True

2.4.2 Random Forest

The Random Forest model is an ensemble method that constructs multiple decision trees and merges their predictions to improve accuracy and control overfitting.

Table 17: Hyperparameters for Random Forest

Hyperparameter	Value
Number of Estimators	100
Max Depth	None
Min Samples Split	2

2.4.3 XGBoost

The XGBoost model was trained on TF-IDF features and evaluated on the validation set, achieving a validation accuracy of 0.8416.

Table 18: XGBoost Model Performance

Metric	**Value**
Validation Accuracy	0.8416

2.4.4 K-Nearest Neighbors (KNN)

KNN is a simple yet effective algorithm that classifies instances based on the majority class among its k nearest neighbors in the feature space.

Table 19: Hyperparameters for KNN

Hyperparameter	Value
Number of Neighbors	5
Weights	Distance

2.4.5 Logistic Regression

Logistic regression is a statistical method for predicting binary classes. It applies the logistic function to model the probability that a given input belongs to a particular category.

2.5 Results

The validation accuracies obtained from our experiments are as follows:

- **Naive Bayes**: 0.8708
- **Random Forest**: 0.8460
- **MLP**: 0.8956
- **KNN**: 0.7824
- **Logistic Regression**: 0.8668

The training process for the MLP included several iterations with loss values decreasing significantly, indicating effective learning.

Table 20: Hyperparameters for Logistic Regression

Hyperparameter	Value
Solver	L-BFGS
C	1.0
Max Iterations	100

2.6 Conclusion

In summary, we demonstrated the effectiveness of various sentiment analysis models on the Stanford movie review corpus. The results indicate the potential for improvement through further experimentation and exploration of alternative approaches.

3 Part C: Custom Naive Bayes

3.1 Introduction

In this section, we explore the Custom Naive Bayes model, designed to classify reviews into positive and negative sentiments based on textual data. The model was trained on a dataset of book reviews, where the scores ranged from 1 to 5. Reviews with scores less than or equal to 2 are labeled as negative (0), while those with scores greater than or equal to 4 are labeled as positive (1). The primary objective is to evaluate the model's effectiveness in accurately predicting the sentiment of book reviews based on the text provided.

3.2 Dataset

The dataset used for this analysis is a collection of book reviews, specifically focusing on the columns 'review/text' and 'review/score'. The dataset was filtered to include only those reviews with scores of 2 or lower (negative sentiment) and scores of 4 or higher (positive sentiment). This filtering helps ensure a balanced dataset with clear distinctions between the two sentiment classes.

3.3 Custom Naive Bayes Model

The Custom Naive Bayes model was implemented to classify reviews based on the frequency of words in the text data. The model is structured as follows:

3.3.1 Model Description

The Naive Bayes algorithm is based on Bayes' theorem, which describes the probability of a class given the features (in this case, the words in the reviews). The "naive" assumption in Naive Bayes is that the features are conditionally independent

given the class label. This simplifies the computation, allowing the model to estimate the probability of each class using the frequency of the words in the training data.

3.3.2 Process Overview

1. Data Preprocessing: - The reviews are filtered based on their scores, creating a binary label for sentiment (positive or negative). - The text data is vectorized using 'CountVectorizer', which transforms the text into a matrix of token counts. This allows the model to analyze the frequency of each word in the reviews.

2. Model Training: - The model estimates the prior probability of each class (the proportion of positive and negative reviews) and the conditional probability of each word given each class (using Laplace smoothing to handle unseen words). - The model is trained using the vectorized training data.

3. Prediction: - For each review in the test set, the model calculates the log probabilities for each class based on the word frequencies, then selects the class with the highest probability.

3.3.3 Features of the Model

- Word Frequency: Utilizes the Count Vectorizer to convert text into a matrix of token counts, enabling the analysis of word occurrences. - Laplace Smoothing: Applied to ensure that no word has a zero probability, which could otherwise skew the results when making predictions.

3.3.4 Results

The performance of the Custom Naive Bayes model was evaluated using accuracy and F1 score metrics. The accuracy reflects the overall percentage of correct predictions, while the F1 score provides a balance between precision and recall, particularly useful in imbalanced datasets.

Custom Naive Bayes Accuracy: 0.8589 Custom Naive Bayes F1 Score: 0.8584

3.3.5 Confusion Matrix

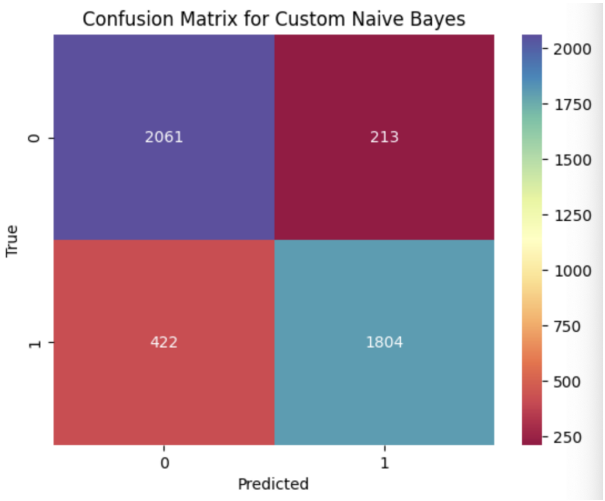

Figure 6: Confusion Matrix

3.3.6 Conclusion

By leveraging word frequency and Laplace smoothing, the model effectively estimated class probabilities, yielding an accuracy of 0.8589 and an F1 score of 0.8584.

For the Naive Bayes models from Part A, we had the accuracies of 0.696, 0.699, and 0.708.

Thus the custom model works better.