Sentiment Analysis of Product Reviews

I. Introduction

Objective:

- Build a machine learning pipeline to analyze the sentiment of product reviews. The goal is to classify each review as positive, negative, or neutral, utilizing a variety of machine learning models and deep learning techniques.

II. Data Preprocessing

- Process of transforming raw data into a clean and usable format before feeding it into a machine learning model.
- For text data, preprocessing can also include removing punctuation, stopwords, and performing tokenization or vectorization.

Step-1: Label Encoding

- The original dataset contained two columns: **"Review"** and **"Sentiment"**. Ratings in the "Sentiment" column range from one to five Stars. But analyze the sentiment as positive, negative, or neutral, the ratings were mapped into three distinct categories. After mapping I changed "Sentiment" column as **"Sentiment Category".**

- **Stars 1 and 2** were categorized as **negative sentiment (-1)**.
- **Stars 3** were categorized as **neutral sentiment (0)**.
- **Stars 4 and 5** were categorized as **positive sentiment (1)**.

Step-2: Handling Missing Values

- I checked for any missing values in the dataset. The "Sentiment Category" field had no missing values.
- There was only one missing value in the "Review" column, So I decided to **drop the corresponding row**.
- **Why Dropping**: In this case, missing value was in the "Review" column, which is the main feature used for sentiment analysis. Since it is not possible to analyze sentiment without the actual review text, it was logical to remove that row to maintain the integrity of the dataset. Since only one row had a missing value, dropping it had a negligible impact on the overall dataset.

Step-3: Text Cleaning

- Words in the reviews are the main focus of sentiment analysis. Punctuation or numbers generally do not contribute to the sentiment classification task. Therefore, I removed all punctuation marks and numeric values from the **"Review"** column.
- This step ensures that the dataset is cleaned of unnecessary symbols, leaving only the relevant text, which is better for accurately analyzing sentiment in the reviews.

Step-4: Removing Duplicates

- To ensure the quality and consistency of the dataset, I first converted all reviews to lowercase. By converting the text to lowercase, I standardized the reviews, ensuring that variations in capitalization do not lead to duplicates being overlooked.
- After standardizing the case, I checked for and removed any duplicate reviews. Duplicate reviews can distort the model's learning process by over-representing certain examples, which could lead to biased results.

Step-5: Tokenize and Vectorize the text data

- After cleaning, the following methods were used to transform the text data into a numerical format in order to prepare it for modeling:

1. Bag of Words (BoW):

- **Why:** Simple and interpretable technique that represents text as the frequency of terms in a document.
- It offers a baseline method for feature extraction, evaluates how well n-gram frequencies capture textual patterns and is widely utilized due to its effectiveness and adaptability in managing stop words, n-grams, and feature constraints.
- **Stop Words Removal**: Since common stop words (e.g., "the," "is," "and") do not contribute significantly to the sentiment analysis task as they are not context-specific. Removing stop words reduces noise and improves model focus on meaningful terms.
- **N-grams**: I considered n-grams ranging from unigrams (single words) to 4-grams to capture context and phrase structures, configured using “ngram_range=(1, 4)”

  Unigrams: represent individual words, providing basic insights.

  Bigrams and Trigrams: capture short phrases and sentiment-related structures like "not good" or "very happy."

  4-grams: help identify slightly longer context without over-complicating the model.

  The range (1,4) was chosen to balance simplicity and context depth.
- **Feature Limitation**: I used "max_features=20000" to limited the maximum number of features to 20,000 in order to control dimensionality and computation.
- Text data often leads to high-dimensional feature spaces, which can:

  Increase computational cost.

  Introduce sparsity that may hinder model performance.

2. TF-IDF (Term Frequency-Inverse Document Frequency):

- A more advanced method to give importance to words that are significant in a document but appear less frequently across other documents.
- TF-IDF enhances the BoW matrix by assigning weights based on term importance:

  Term Frequency (TF): ensures that commonly occurring terms within a document are emphasized.

  Inverse Document Frequency (IDF): down-weights terms that appear across many documents (e.g., "good" or "product"), as these are less useful for distinguishing sentiments.
- **Why:** This transformation mitigates the problem of frequently occurring yet uninformative words dominating the feature space.

III. Model Training and Parameter Choices

Step-1: Data Splitting

- I splitted dataset into training (80%) and testing (20%) subsets to evaluate the model's generalization capability on unseen data.
- A random state was used to ensure reproducibility of results.
- By reserving 20% of the data for testing, it achieve a balance between training the model effectively and assessing its performance on data it hasn’t seen before.
- Using a fixed random state means the data is split the same way every time I run my code. This makes it easier to compare different models fairly because they all use the same data setup.

Step-2: Model Training

- For model training I used three different models were selected to explore various approaches to sentiment classification.

1. Logistic Regression

- Approach: Ordinal Logistic Regression using the mord library.
- Why Logistic Regression?

  Logistic regression is simple, efficient, and interpretable, making it a baseline model for classification tasks.
- Using ordinal logistic regression allows the model to consider the inherent order in sentiment categories (-1 < 0 < 1), potentially improving performance for ordinal data.

2. Naïve Bayes

- Approach: Multinomial Naive Bayes.
- Why Naive Bayes?

  Naive Bayes is well-suited for text classification tasks due to its probabilistic nature and ability to handle high-dimensional data like text vectors.
- It is computationally efficient and often performs well.

3. Support Vector Machine (SVM)

- Approach: SVM with various kernels and hyperparameters, optimized using GridSearchCV.
- Why SVM?

  SVM is effective in handling high-dimensional spaces like TF-IDF vectors.
- It can model complex decision boundaries with different kernels (linear, RBF, polynomial), providing flexibility.

- SVM is robust to overfitting, especially in cases where the number of features exceeds the number of samples.

Step-3: Hyperparameter Tuning

- For the SVM model, GridSearchCV was implemented to find the optimal combination of hyperparameters:
- Kernel: [Linear, RBF, and polynomial]
- C: [0.1, 1.5] (controls the trade-off between achieving a low error on the training data and minimizing the margin's width).
- Gamma: ['scale', 'auto'] (defines how far the influence of a single training example reaches, used for non-linear kernels like RBF).
- Best Hyperparameters for SVM were found to be C=1, gamma='scale' and kernel='rbf'.
- Cross-validation was performed to ensure the best hyperparameters were selected based on model performance on validation splits.
- Grid search systematically evaluates all combinations of hyperparameters, ensuring the best configuration for SVM is selected.

IV. Results and Analysis

Step-1: Performance Metrics

- I evaluated model performance using Performance Metrics which is used for Classification tasks.

  Accuracy: The proportion of correctly classified samples.

  Precision: The ratio of true positive predictions to the total predicted positives.

Recall: The ratio of true positive predictions to the actual positives in the data.

F1 Score: The harmonic mean of precision and recall, balancing the two metrics.

Confusion Matrix: Provides insight into the classification results by showing the number of true positives, false positives, true negatives, and false negatives.

Step-2: Model Performance Summary

- The table below summarizes the performance of the three models:

	Logistic			SVM			NB		
	precision	recall	f1-score	precision	recall	f1-score	precision	recall	f1-score
-1	0.79967	0.712255	0.753435	0.721535	0.857353	0.783602	0.709091	0.860294	0.777409
0	0.30124	0.437500	0.356803	0.481481	0.137712	0.214168	0.631579	0.012712	0.024922
1	0.79896	0.714729	0.754501	0.730787	0.840310	0.781731	0.683299	0.856331	0.760092
accuracy	0.66050	0.660500	0.660500	0.712543	0.712543	0.712543	0.696076	0.696076	0.696076

Step-3: Comparative Analysis of Classification Models

1. Logistic Regression:

- Logistic Regression is strong in precision.
- Consistently performs well across precision, recall, and F1-score for category -1 and 1, but it performs poorly on category 0.

2. Support Vector Machine (SVM):

- SVM is the best model overall, particularly for category -1 and 1, with the highest recall for category -1 and a good balance of precision and recall for category 1.
- Performs well for category 1 with a higher F1-score than Logistic Regression.

- Struggles with category 0, having a very low recall (0.1377), which drastically lowers its F1-score for this category.

3. Naive Bayes

- Struggles with category 0, achieving a very low recall (0.0127), which severely affects its F1-score.
- For category -1 and category 1, Naive Bayes performs similarly to SVM, with high recall but lower precision than Logistic Regression, leading to a less balanced performance across all classes.

- For an overall robust model across all classes, SVM would be the most reliable choice, especially if class 0 is not the focus of the analysis.
- The neutral class performed weakly across all models because it often contains a mix of positive and negative sentiments, making it harder to classify accurately.

  For example, sentences like "Good sound, but controls are hard to use" or "It looks cheaply made, but it works" express both pros and cons. This ambiguity confuses the models, as they struggle to identify clear patterns for neutral reviews. Additionally, neutral samples may overlap with positive or negative classes, further reducing classification accuracy.

Step-4: Visualization

- To better illustrate the results, the following plots were created:
- **Confusion Matrices** for each model to visualize classification performance.
- **Bar Graph** comparing the Accuracy across the three models.

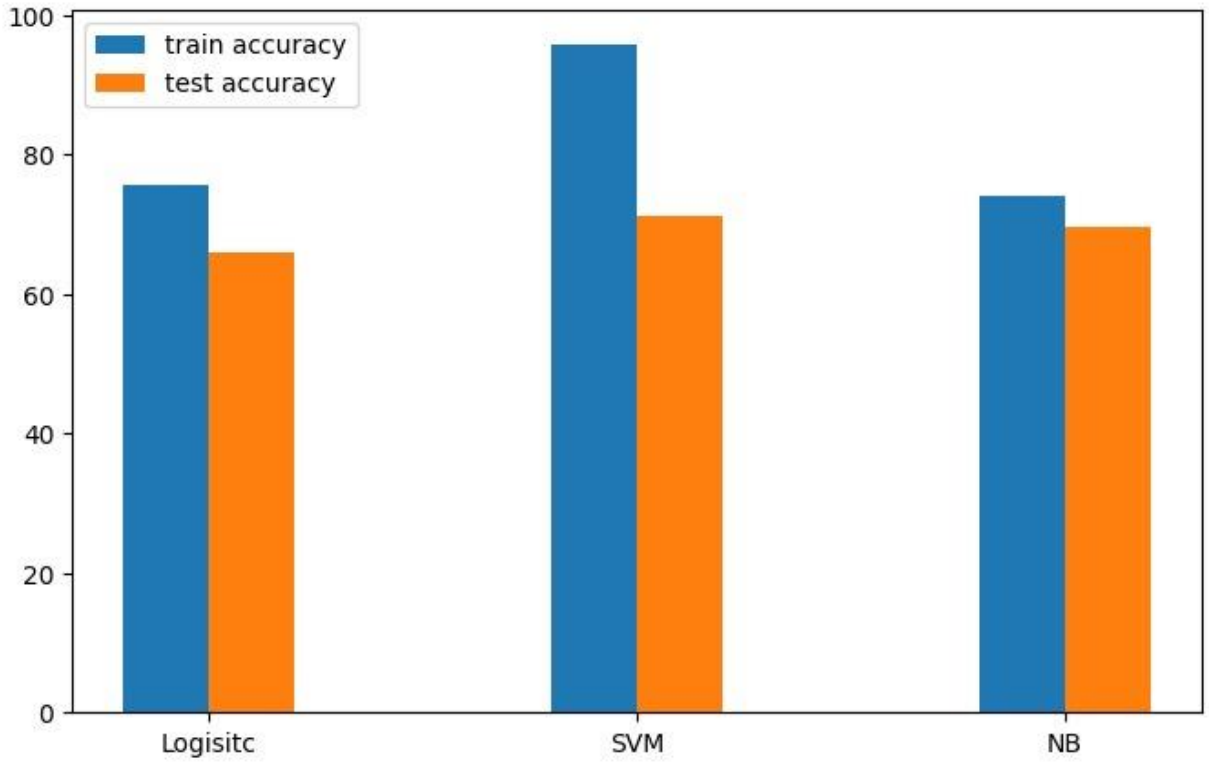

Figure 1: Bar Graph

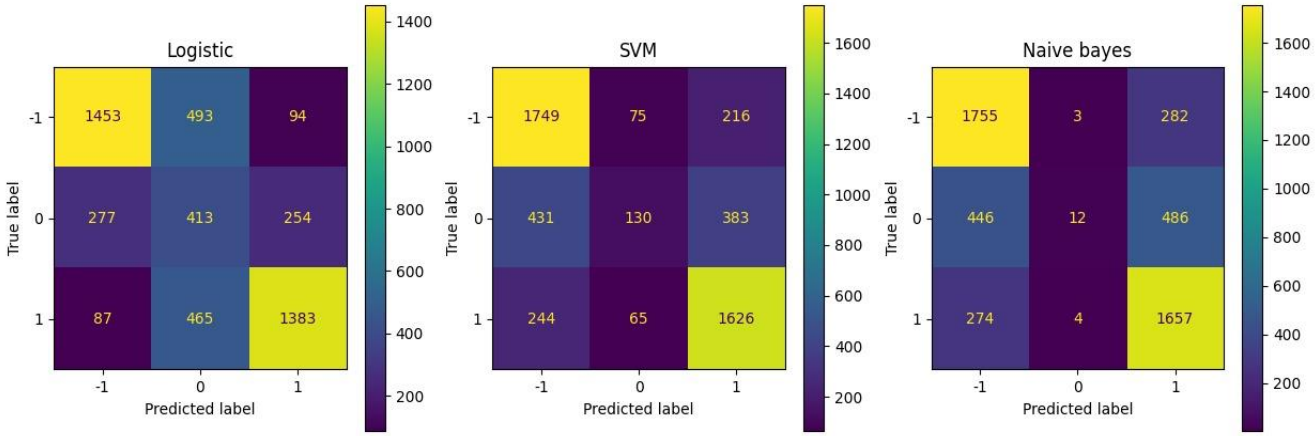

Figure 2: Confusion Matrix