NLP 220 ASSIGNMENT 2

Shriya Sravani Y
UCSC
sy4@ucsc.edu

1 Introduction

This report is divided into two parts, the first part focuses on implementing multiclass classification on an e-commerce dataset, using feature engineering techniques on SVM(Support Vector Machine), Logistic Regression and Decision Tree classifiers. Nine such models are made with the aim to find the accuracies, macro F1 scores and their training and inference times to determine the best model.

Then hyperparameter exploration is done on these models to find the best fits. The latter part of this report gives OneVsRest accuracies and macro F1-scores.

2 Part A - Feature Engineering for SVM, Logistic Regression and Decision Trees

2.1 Dataset

The dataset, named **ecommerceDataset.csv,** contains text-based descriptions of items and corresponding category labels.The dataset has 50424 entries and two columns. These columns are renamed as 'category' and 'description' for ease of understanding. The 4 classes are household, books, electronics and clothes and accessories.

After loading the dataset, I split it into three sets for training, validation, and testing, as follows:

- Training Set: 70% of the data
- Validation Set: 10% of the data
- Test Set: 20% of the data

To ensure reproducibility, I used a fixed random seed for data shuffling during the split. Below is the class distribution of the given data.

I utilized the train_test_split method from sklearn.model_selection to create the splits. This split allows the model to be trained and validated effectively, with the test set providing an unbiased evaluation of performance. The validation set was used for hyperparameter tuning and early stopping if required.

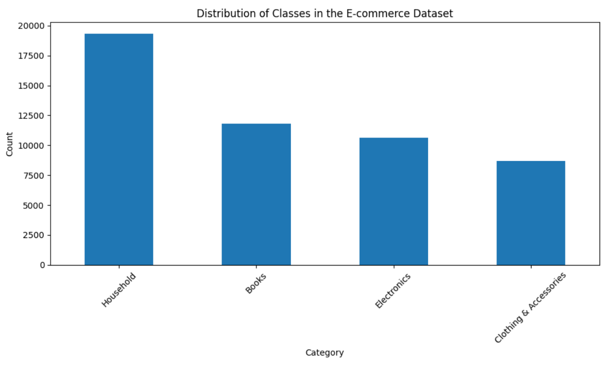


Figure 1: Distribution of Review Scores

2.2 Feature Engineering

The feature engineering techniques used are:

1. **BOW**

BOW converts text into numerical features based on the occurrence of words within a document. Each unique word in the dataset forms a feature, and the values represent the count of occurrences of each word in the document. By converting text data into numerical vectors, BOW enables machine learning algorithms to interpret and learn from the data.

The CountVectorizer with ngram_range=(1, 2) is used to extract features from the text data. It generates both unigrams and bigrams, with a maximum of 10,000 features.

2. **Tf-Idf**

TF-IDF is a text representation technique used to convert textual data into numerical vectors for machine learning. It calculates the importance of a word in a document relative to its frequency across the entire dataset. The TF (Term Frequency) measures how frequently a word appears in a document, while IDF (Inverse Document Frequency) downscales words that occur frequently across many documents, as they are often less informative.

3. **N-Gram**

An n-gram is a contiguous sequence of 'n' items from a given text. Using n-grams as features allows machine learning models to capture more context and relationships between words. In this model, bi-grams (1-gram and 2-gram combinations) are used to capture both single words and pairs of consecutive words, enriching the feature set for the classifier.

Classifier	Feature Engineering	Accuracy	Macro F1 Score
SVM	Count Vectorizer (BoW)	0.96	0.96
SVM	TF-IDF	0.96	0.97
SVM	N-grams	0.96	0.96
Logistic Regression	Count Vectorizer (BoW)	0.96	0.96
Logistic Regression	TF-IDF	0.94	0.94
Logistic Regression	N-grams	0.96	0.95
Decision Tree	Count Vectorizer (BoW)	0.94	0.95
Decision Tree	TF-IDF	0.95	0.95
Decision Tree	N-grams	0.94	0.94

Table 1: Comparison of Accuracy and Macro F1 Score for Each Model and Feature Engineering Technique

Support Vector Machines (SVM) are known for their effectiveness in high-dimensional spaces, which aligns well with text data after vectorization. Here's how SVM performed across the three feature engineering techniques:

BoW: The accuracy and F1 score is 0.96. The SVM model with BoW performed well, achieving high accuracy and balanced F1 scores across classes.

TF-IDF: The accuracy is 0.96 and macro F1 score is 0.97. The combination of SVM with TF-IDF was slightly more effective than BoW. TF-IDF emphasizes rare but meaningful words, resulting in the highest accuracy and F1 score for the SVM models.

N-grams: The accuracy and macro F1 score is 0.96. The SVM model with N-grams performed similar to BoW, as N-grams capture additional word sequences, but not full semantic relationships.

TF-IDF was the best-performing feature technique for SVM, likely due to its weighting of less frequent but important words, giving SVM a nuanced representation for classification.

Logistic Regression is a linear model that often works well on text classification tasks, especially with feature engineering methods like BoW or TF-IDF.

BoW: The accuracy is 0.96 and macro F1 Score is 0.96. Logistic Regression with BoW achieved the highest accuracy among all models, performing exceptionally well across classes.

TF-IDF: The accuracy is 0.94 and macro F1 Score is 0.94. The accuracy for Logistic Regression with TF-IDF was still high, but the Macro F1 Score was slightly lower.

N-grams: The accuracy is 0.96 and macro F1 Score is 0.95. The inclusion of bi-grams adds some contextual information, improving classification where adjacent words provide additional category clues.

Count Vectorizer was the best feature engineering technique for Logistic Regression, achieving the highest accuracy and a high Macro F1 score. This may be due to Logistic Regression's effectiveness in distinguishing linearly separable categories based on word frequency.

Decision Trees are known for their interpretability but can struggle with high-dimensional sparse data like text. Despite these challenges, Decision Tree models provided reasonable performance with certain feature engineering techniques.

BoW: The accuracy is 0.94 and macro F1 Score is 0.95. Decision Tree with BoW had lower accuracy compared to SVM and Logistic Regression.

TF-IDF: The accuracy is 0.95 and macro F1 Score is 0.95. The Decision Tree performed slightly better with TF-IDF than with BoW.

N-grams: The accuracy is 0.94 and macro F1 Score is 0.94. Decision Tree with N-grams did not show significant improvement over BoW or TF-IDF. Although N-grams add context, the Decision Tree may have struggled to effectively split based on high-dimensional sequences of words, leading to performance on par with other techniques.

Overall, the **SVM model** with **TF-IDF** feature engineering has the highest overall performance with:

- Accuracy: 0.96
- Macro F1 Score: 0.97

SVM is known for maximizing the margin between classes, thereby making them effective in finding a clear decision boundary. It effectively addresses the class imbalance by focusing on each class seperately and improves the F1 score. Coupled with the TF-IDF that captures the importance of terms relative to each document and reduces the noise in feature space, this model gives the best overall accuracy and macro F1 score.

Training and Inference Time

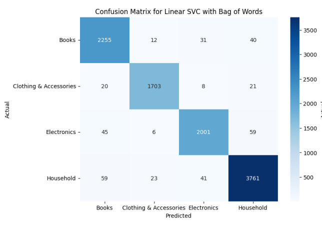
The training and inference times for the models are given in the table below.

Classifier	Feature Engineering	Training Time (s)	Inference Time (s)
SVM	Count Vectorizer (BoW)	49.9331	0.0065
SVM	TF-IDF	1.2742	0.0036
SVM	N-grams	59.4679	0.0074
Logistic Regression	Count Vectorizer (BoW)	20.619	0.013
Logistic Regression	TF-IDF	8.4000	0.0069
Logistic Regression	N-grams	25.4000	0.0085
Decision Tree	Count Vectorizer (BoW)	19.5664	0.0085
Decision Tree	TF-IDF	25.3899	0.0099
Decision Tree	N-grams	17.4915	0.0082

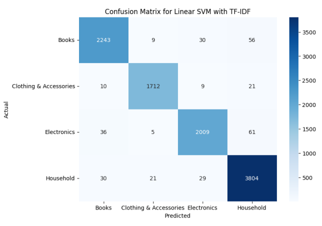
Table 2: Comparison of Training and Inference Time for Each Model and Feature Engineering Technique

The SVM with TF-IDF has the shortest training time (1.27 seconds) and the shortest inference time (0.0036 seconds), making it the most efficient model in terms of computational cost and suitable for applications where time efficiency is critical.

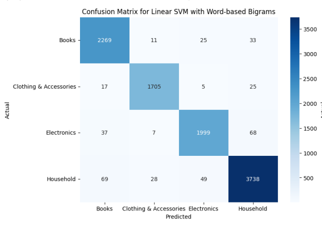
Confusion Matrices



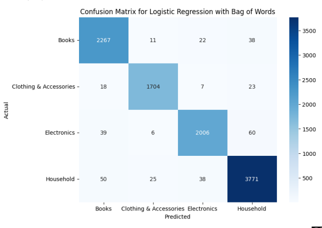
(a) Linear SVC with Bag of Words



(b) Linear SVC with TF-IDF

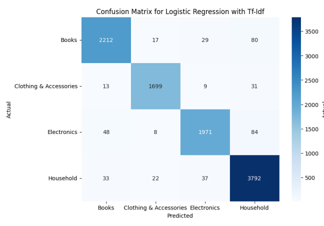


(c) Linear SVC with Bigrams

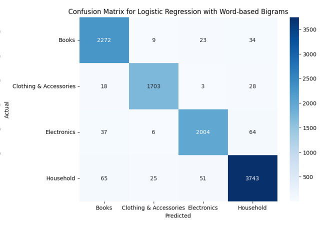


(d) Logistic Regression with Bag of Words

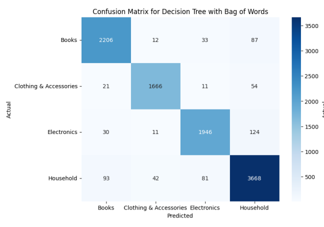
Linear SVC consistently outperforms Logistic Regression and Decision Trees across all feature types. It has the least number of misclassifications and handles class separations better. TF-IDF generally gives the best results across all models. It reduces misclassifications by assigning appropriate weights to important terms, improving separability between classes. Word-based Bigrams adds additional context compared to BoW and TF-IDF, but its performance improvement is minimal.



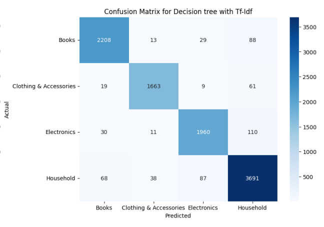
(a) Logistic Regression with TF-IDF



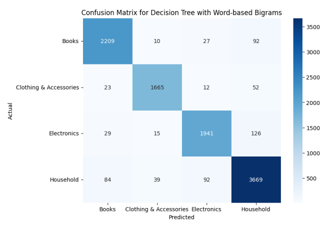
(b) Logistic Regression with Bigrams



(c) Decision Tree with Bag of Words



(d) Decision Tree with TF-IDF



(e) Decision Tree with Bigrams

Hyperparameters

Classifier	Feature Engineering	Accuracy
SVM	Count Vectorizer (BoW)	0.9649
SVM	TF-IDF	0.9691
SVM	N-grams	0.9627
Logistic Regression	Count Vectorizer (BoW)	0.9668
Logistic Regression	TF-IDF	0.9684
Logistic Regression	N-grams	0.9646
Decision Tree	Count Vectorizer (BoW)	0.9379
Decision Tree	TF-IDF	0.9350
Decision Tree	N-grams	0.9357

Table 3: Comparison of Accuracies for Each Model and Feature Engineering Technique after Grid Search

Grid search is an exhaustive search technique for hyperparameter tuning that evaluates every possible combination of a given set of parameters to find the configuration that results in the best model performance based on a specified evaluation metric.

Linear Support Vector Machines (SVMs) are widely used for classification tasks, especially with high-dimensional data. The most critical hyperparameter in a linear SVM is the regularization parameter ***C***. he regularization parameter C, controls the trade-off between achieving a low training error and a low testing error. The type of kernel function used to compute the decision boundary.

For linear SVMs, only the '**linear'** kernel is used, so this parameter is fixed here.

C in **Logistic Regression** is a regularization parameter that controls the strength of regularization.**Penalty** determines the type of regularization, i.e, 'l2': L2 regularization (Ridge), which adds a penalty proportional to the square of the coefficients. It is commonly used and effective in reducing overfitting. 'l1': L1 regularization (Lasso), which adds a penalty proportional to the absolute values of the coefficients, encouraging sparsity in the model. Solver is the optimization algorithm used to fit the model. Here, **'liblinear'** is chosen because it supports both L1 and L2 regularization and is effective for smaller datasets or datasets with a high number of features.

For **Decision Trees**, the hyperparameters used are:

max_depth: The maximum depth of the tree. Limiting depth helps prevent overfitting by controlling how many levels the tree can grow. Setting this to None allows nodes to expand until all leaves are pure or contain fewer than the minimum number of samples specified by min_samples_split.

min_samples_split: The minimum number of samples required to split an internal node.

min_samples_leaf: The minimum number of samples that a node must have to be considered a leaf. Larger values help smooth the model by creating broader leaves, which can help with generalization.

criterion: The function used to evaluate the quality of a split:

-'gini': Gini impurity, which measures the impurity of the nodes.

-'entropy': Entropy, based on information gain, which measures the randomness in the split. Both can work well; often, their performance depends on the dataset.

3 Part B - OneVsRest Exploration

OneVsRest (OvR) is a strategy used in multiclass classification where the problem is decomposed into multiple binary classification tasks. Each binary task involves isolating one class as the "positive" class and grouping all other classes as the "negative" class. This approach is particularly useful when the classifier natively supports only binary classification, as it allows such classifiers to be extended to handle multiclass problems.

- For each class C_i, a separate binary classifier is trained.
- During training, samples belonging to C_i are labeled as positive, while all other samples are labeled as negative.
- For prediction, the classifier computes probabilities (or scores) for each class, and the class with the highest probability is assigned as the final prediction.

Results from the Task

- **Individual Macro F1 Scores:**
  - Books: 0.9797
  - Clothing and Accessories: 0.9819
  - Electronics: 0.9714
  - Household: 0.9694
- **Average Macro F1 Score:** 0.9756

The **Macro F1 Score** for each class measures the balance between precision and recall when isolating that class versus the rest. The average score of 0.9756 indicates high overall performance in the multiclass classification task.

3.1 Interpretation of Precision-Recall and ROC Curves

Precision-Recall Curves

- Precision-Recall curves, plotted for each class, represent the trade-off between precision (positive predictive value) and recall (sensitivity) across different thresholds.
- Observation:
  - The curves exhibit high precision and recall across all classes, indicating that the classifier is effective at correctly identifying positives without many false positives or false negatives.
  - The steep drop at the end of each curve occurs due to thresholds that are too lenient, leading to over-predictions.

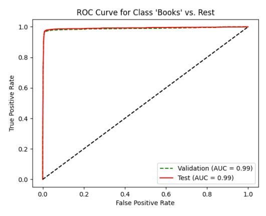
ROC Curves

- ROC (Receiver Operating Characteristic) curves show the trade-off between the true positive rate (TPR) and false positive rate (FPR) for varying thresholds.

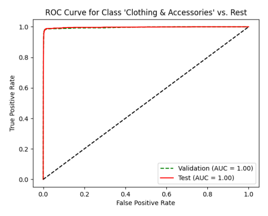
- Observation:
  - The ROC curves for both validation and test sets are almost identical for each class, with areas under the curve (AUC) close to 1.0. This indicates strong model performance and little variance between validation and test predictions.

The individual F1 scores for each class are high, showing that the classifier maintains a good balance between precision and recall. The slight variance in F1 scores across classes may be attributed to the distribution and characteristics of the dataset (e.g., imbalanced classes or overlapping features). The near-identical ROC curves for validation and test sets suggest that the model generalizes well and avoids overfitting.

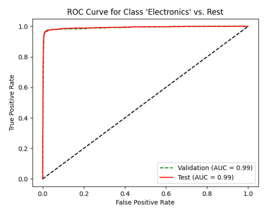
ROC and Precision-Recall Curves



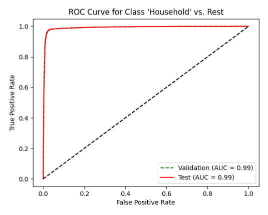
(a) ROC for Books (AUC = 0.99)



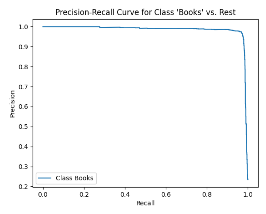
(b) ROC for Clothing (AUC = 1.00)



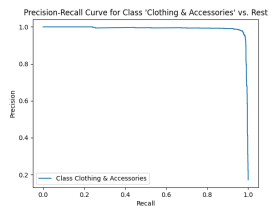
(c) ROC for Electronics (AUC = 0.99)



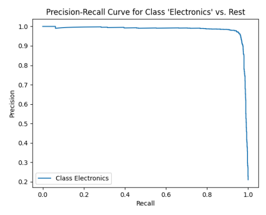
(d) ROC for Household (AUC = 0.99)



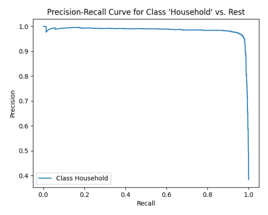
(e) PR Curve for Books



(f) PR Curve for Clothing



(g) PR Curve for Electronics



(h) PR Curve for Household

3.2 Conclusion

The OneVsRest strategy demonstrates a good performance for the multiclass classification task, as reflected in both the quantitative metrics and graphical representations. The high ROC and Macro F1 scores highlight the classifier's effectiveness across all classes. These results suggest that the chosen model and feature engineering approach are well-suited for this problem.