NLP 220 Homework 2

Mudit Arora

November 9, 2024

1 Feature Engineering for SVM, Logistic Regression, and Decision Tree

1.1 Introduction

In this first problem, we'll implement features that can be used with the Support Vector Machine, Logistic Regression, and Decision Tree classifiers implemented in sci-kit-learn.

1.2 Dataset

Table 1: Data Columns Summary

#	Column	Non-Null Count	Dtype
0	Category	50424	object
1	Description	50424	object

We're provided with an E-Commerce dataset of `ecommerceDataset.csv`.

The dataset contains Descriptions of :

- Category:
- Description:

1.3 New Feature

We are required to build at least three different distinct features that process the Description column.

1.3.1 Length of Description

The `Description_Length` feature calculates the word count for each description, capturing text length as a numerical feature. This is important because description length can indicate description depth or verbosity, which may correlate with sentiment or relevance in text analysis.

	Category	Description	Description_Length
0	Household	SAF 'Floral' Framed Painting (Wood, 30 inch x ...	59
1	Household	SAF 'UV Textured Modern Art Print Framed' Pain...	224
2	Household	SAF Flower Print Framed Painting (Synthetic, 1...	184
3	Household	Incredible Gifts India Wooden Happy Birthday U...	184
4	Household	Pitaara Box Romantic Venice Canvas Painting 6m...	230

Figure 1: Length of Description Output

1.3.2 Sentiment Polarity

Sentiment polarity is useful in classification because it reveals the emotional tone in product descriptions, which can help differentiate categories based on consumer appeal and perception.

	Category	Description	Description_Length
0	Household	SAF 'Floral' Framed Painting (Wood, 30 inch x ...	59
1	Household	SAF 'UV Textured Modern Art Print Framed' Pain...	224
2	Household	SAF Flower Print Framed Painting (Synthetic, 1...	184
3	Household	Incredible Gifts India Wooden Happy Birthday U...	184
4	Household	Pitaara Box Romantic Venice Canvas Painting 6m...	230

Figure 2: Sentiment Polarity Output

1.3.3 Bag of Words (BoW)

We used Bag of Words as our 3rd feature since it can be useful to offer insights to identify commonly used terms and themes within a dataset. It helps identify patterns in word usage across different product types. BoW is simple yet effective, especially when category.

	10	100	11	12	13	14	15	16	18	19	...	written	year	years	yellow	yes	yet	york	you	your	yourself
0	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	2	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	3	0	0
2	0	0	0	0	1	0	0	0	0	0	...	0	0	0	0	0	0	0	2	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	7	6	0
4	0	0	0	0	0	1	0	0	0	0	...	0	0	0	0	0	0	0	0	2	0

Figure 3: Frequency/Bag of Words

1.3.4 Term Frequency-Inverse Document Frequency (TFIDF)

TFIDF is being as our 4th feature since it emphasizes important words that uniquely define categories by reducing the impact of commonly used words. Unlike BoW, TF-IDF highlights terms that appear frequently in specific categories but are less common across other categories, improving classification accuracy by focusing on distinctive vocabulary.

	10	100	1000	11	12	13	14	15	16	17	...	writer	writing	written	year	years	yellow	yes	york	young
0	0.125199	0.0	0.0	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.000000	0.0	0.0	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.000000	0.0	0.0	0.0	0.0	0.122157	0.000000	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.000000	0.0	0.0	0.0	0.0	0.000000	0.000000	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.000000	0.0	0.0	0.0	0.0	0.000000	0.089177	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

5 rows × 1000 columns

Figure 4: TFIDF

1.4 Data Preprocessing

The data preprocessing steps involved:

- Create a multi-class classification dataset: We converted the 4 string labels: Household, Books, Clothing and Accessories, and Electronics into 0, 1, 2, and 3, respectively, for easier use.
- Creating a train/test split using the `train_test_split` method from `sklearn.model_selection`, ensuring the data is shuffled. Utilize a fixed random seed of your choice for reproducibility, allocating 70% of the data for training, the remaining 10% for validation, and the rest 20% for testing.

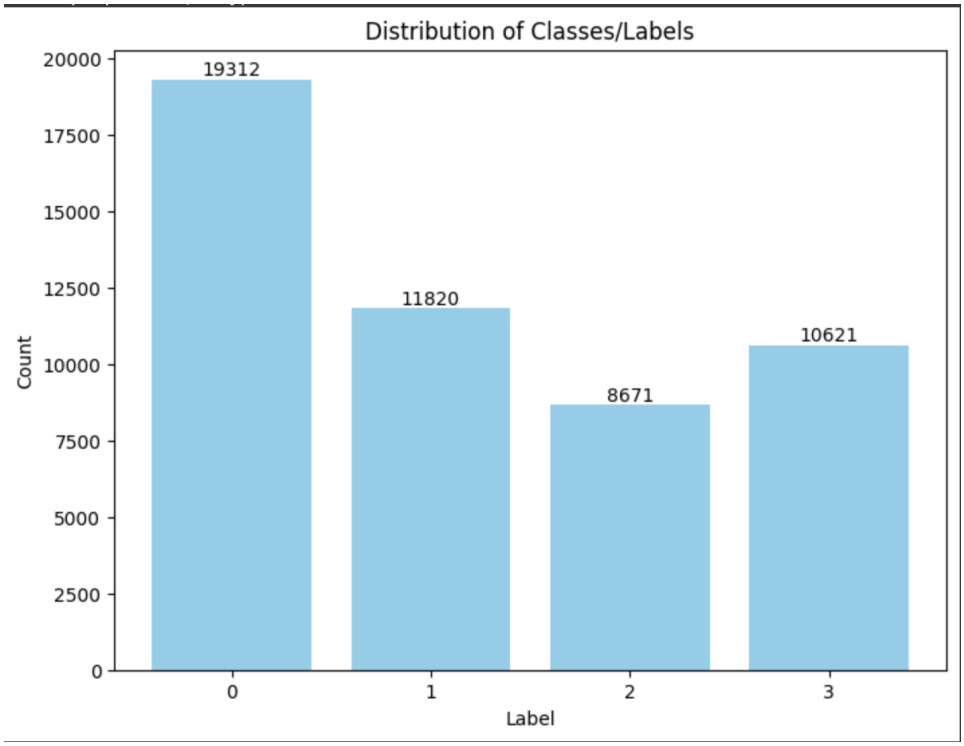

Figure 5: Distribution of Class labels

1.5 Feature Engineering

We tried experimenting by combining different features with each other and decided to go with the those that gave good accuracy.

- **1st. Combination of all the features**: The initial feature engineering process combines multiple features, including `Description_Length`, `Sentiment`, word frequency counts (from Bag of Words), and `TF-IDF` values, to create a comprehensive feature set. captures both the structure and the meaning of the product descriptions. This allows the model to consider various aspects of text, such as length, tone, common keywords, and unique terms, improving classification accuracy.

- **2nd. Description of Length + Sentiment + TFIDF**: This feature captures both the structure and emotional tone of product descriptions. `Description_Length` provides a basic structural feature, while `Sentiment` captures the emotional tone, and `TF-IDF` highlights important terms. Together, these features help the

model differentiate categories more effectively, while the data split ensures unbiased training, validation, and testing.

- **3rd. Description of Length + Sentiment + BoW**: This feature engineering approach capture both structural and semantic information from product descriptions. `Description_Length` provides insight into the length of the descriptions, while `Sentiment` captures the emotional tone. The word frequency counts help identify the most common terms in the descriptions, providing additional context to distinguish categories. Together, these features improve the model's ability to differentiate between product types based on both their structure and content.

1.5.1 1st Feature Engineering

```
Accuracy: 0.95
              precision    recall  f1-score   support

           0       0.94      0.95      0.95      3906
           1       0.94      0.95      0.95      2356
           2       0.97      0.96      0.96      1730
           3       0.94      0.93      0.93      2094

    accuracy                           0.95     10086
   macro avg       0.95      0.95      0.95     10086
weighted avg       0.95      0.95      0.95     10086

[[3723   73   35   75]
 [  79 2232   14   31]
 [  32   22 1667    9]
 [ 107   39   10 1938]]
```

Figure 6: 1st Support Vector Machine Model

```
Accuracy: 0.94
              precision    recall  f1-score   support

           0       0.93      0.94      0.93      3906
           1       0.94      0.95      0.95      2356
           2       0.96      0.94      0.95      1730
           3       0.93      0.92      0.92      2094

    accuracy                           0.94     10086
   macro avg       0.94      0.94      0.94     10086
weighted avg       0.94      0.94      0.94     10086

[[3682   86   42   96]
 [  83 2230   11   32]
 [  73   21 1622   14]
 [ 133   24   16 1921]]
```

Figure 7: 1st Decision Tree Model

Comparing the accuracy and F1 score of all 3 models, the Support Vector Machine comes out to be the best. For this case, we used LinearSVC() as the classifier.

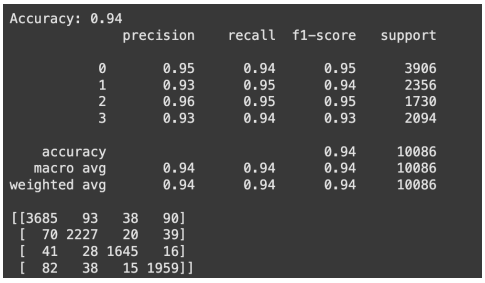

Figure 8: 1st Losgitic Regression Model

1.5.2 2nd Feature Engineering

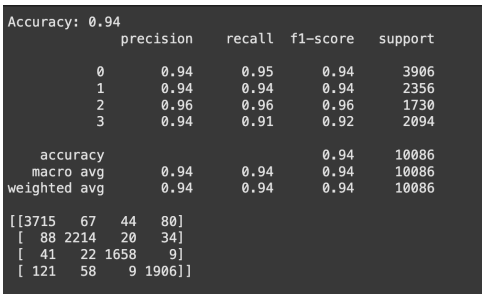

Figure 9: 2nd Support Vector Machine Model

Since all the models have the same accuracies and F1 score, it's best to compare the confusion matrix. On adding all the True Positives (the correct labels), SVM seems to have the most corrected labels.

1.5.3 3rd Feature Engineering

Since both the accuracy and F1 score of both SVM and Decision Tree are same, we'll compare the True Positives from the confusion matrix. On calculating, Decision Tree seem to be a better model for this feature engineering.

1.6 Training/Inference time comparison

1.7 Hyper-parameter Exploration

Among the 3 feature engineering, the 1st one gave the highest accuracy, F1 score, and the most corrected labels. Hence, we'll be using the same feature engineering for the hyper-parameter exploration.

```
Accuracy: 0.94
              precision    recall  f1-score   support

           0       0.94      0.95      0.94      3906
           1       0.94      0.94      0.94      2356
           2       0.96      0.96      0.96      1730
           3       0.94      0.91      0.92      2094

    accuracy                           0.94     10086
   macro avg       0.94      0.94      0.94     10086
weighted avg       0.94      0.94      0.94     10086

[[3715   67   44   80]
 [  88 2214   20   34]
 [  41   22 1658    9]
 [ 121   58    9 1906]]
```

Figure 10: 2nd Decision Tree Model

```
Accuracy: 0.94
              precision    recall  f1-score   support

           0       0.93      0.94      0.94      3906
           1       0.94      0.95      0.95      2356
           2       0.96      0.93      0.94      1730
           3       0.94      0.92      0.93      2094

    accuracy                           0.94     10086
   macro avg       0.94      0.94      0.94     10086
weighted avg       0.94      0.94      0.94     10086

[[3679   83   53   91]
 [  82 2245   11   18]
 [  73   27 1615   15]
 [ 121   37   12 1924]]
```

Figure 11: 2nd Logistic Regression Model

On trying several hyper-parameter tuning such as GridSearchCV, HalvingGridSearchCV, etc. The best that worked for this feature was RandomizedSearchCV.

1.7.1 RandomizedSearchCV

RandomizedSearchCV is a hyperparameter tuning method that randomly samples from a predefined parameter distribution, testing a fixed number of combinations. Unlike GridSearchCV, which exhaustively tries all possible parameter combinations, RandomizedSearchCV is faster and covers a broader range by sampling randomly, making it ideal when tuning over large parameter spaces or with limited computational resources. It's especially useful for efficiently finding near-optimal hyperparameters in less time.

1. **Parameter Space Definition**: The tuning parameters for `LinearSVC`, `Decision Tree Classifier`, and `Logistic Regression` are limited to a simplified range, as shown in the table below:

2. **RandomizedSearchCV Initialization**: `RandomizedSearchCV` is set up with the `LinearSVC`, `Decision Tree Classifier`, and `Logistic Regression`

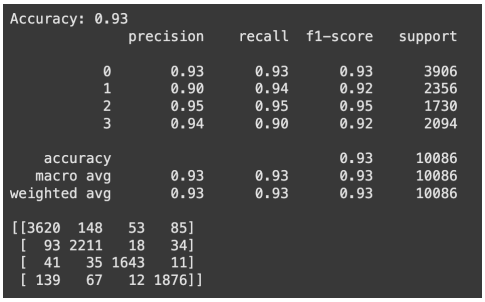

Figure 12: 3rd Support Vector Machine Model

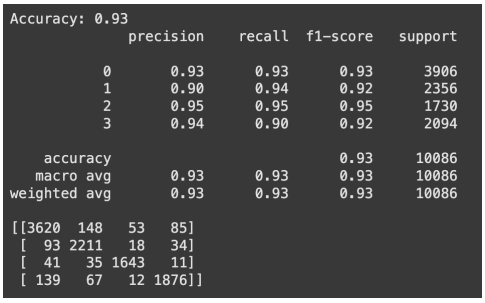

Figure 13: 3rd Decision Tree Model

estimator, using the above parameter space, 5 random iterations (`n_iter=5`), 2-fold cross-validation (`cv=2`), and accuracy as the scoring metric. It uses all available CPU cores (`n_jobs=-1`) and provides a brief output of the tuning progress (`verbose=1`).

3. **Model Fitting**: The search fits the model to the training data (`X_train1` and `y_train1`), testing different parameter combinations to find the best fit.

4. **Validation Set Evaluation**: The tuned model makes predictions on the validation set (`X_val1`), and the accuracy and classification report are displayed, showing performance metrics such as precision, recall, and F1-score.

5. **Save and Test the Best Model**: The best model is saved for final testing. It then makes predictions on the test set (`X_test1`), providing accuracy and a classification report, which offers a final assessment of model performance.

These are the results on the Validation and Test Set:

1.8 OneVsRest Exploration

This analysis implements a multiclass classification approach using the One-vs-Rest (OvR) strategy with Logistic Regression as the base classifier. The OvR approach

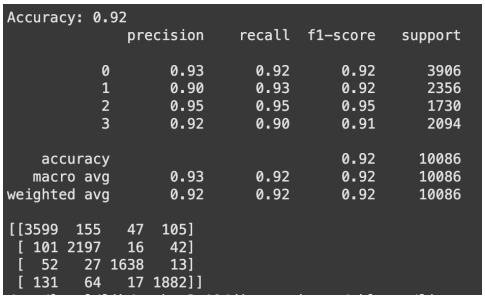

Figure 14: 3rd Logistic Regression Model

Table 2: Training/Inference time

Model #	Model Name	Time (in s)
1	SVM	41.00
1	Decision Tree	26.69
1	Logistic Regression	39.56
2	SVM	2.75
2	Decision Tree	15.12
2	Logistic Regression	20.82
3	SVM	10.88
3	Decision Tree	16.08
3	Logistic Regression	22

Table 3: SVM Parameters

Parameter	**Values**
`C`	[0.1, 1.0, 10.0, 100.0]
`loss`	`squared_hinge`
`tol`	[1e-3, 1e-4]
`max_iter`	1500
`dual`	`True`
`class_weight`	`balanced`

transforms the multiclass classification problem into multiple binary classification problems, one for each class.

1.8.1 Implementation Details

The classification process involves several key steps:

```
Validation Set Performance:
Accuracy: 0.3843712812376041

Classification Report:
              precision    recall  f1-score   support

           0       0.38      1.00      0.56      1938
           1       0.00      0.00      0.00      1196
           2       0.00      0.00      0.00       850
           3       0.00      0.00      0.00      1058

    accuracy                           0.38      5042
   macro avg       0.10      0.25      0.14      5042
weighted avg       0.15      0.38      0.21      5042
```

Figure 15: Val Accuracy on SVM

```
Validation Set Performance:
Accuracy: 0.40420468068226895

Classification Report:
              precision    recall  f1-score   support

           0       0.90      0.22      0.36      1938
           1       0.28      0.96      0.44      1196
           2       0.85      0.45      0.59       850
           3       0.96      0.07      0.12      1058

    accuracy                           0.40      5042
   macro avg       0.75      0.43      0.38      5042
weighted avg       0.76      0.40      0.37      5042
```

Figure 16: Val Accuracy on Logistic Regression

```
Validation Set Performance:
Accuracy: 0.3843712812376041

Classification Report:
              precision    recall  f1-score   support

           0       0.38      1.00      0.56      1938
           1       0.00      0.00      0.00      1196
           2       0.00      0.00      0.00       850
           3       0.00      0.00      0.00      1058

    accuracy                           0.38      5042
   macro avg       0.10      0.25      0.14      5042
weighted avg       0.15      0.38      0.21      5042
```

Figure 17: Val Accuracy on Decision Tree

```
Test Set Performance:
Accuracy: 0.9448740828871703

Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.94      0.95      3906
           1       0.94      0.94      0.94      2356
           2       0.95      0.97      0.96      1730
           3       0.93      0.94      0.93      2094

    accuracy                           0.94     10086
   macro avg       0.94      0.95      0.95     10086
weighted avg       0.94      0.94      0.94     10086
```

Figure 18: Test Accuracy on SVM

Parameter	Values
max_depth	3, 5, 7, 10, None
min_samples_split	2, 4, 6, 8
min_samples_leaf	1, 2, 3, 4
criterion	gini, entropy, log_loss
splitter	best, random
max_features	auto, sqrt, log2

Table 4: Decision Tree Classifier Hyperparameters

Parameter	Values
C	0.1, 1.0, 10.0
solver	lbfgs
max_iter	1000

Table 5: Logistic Regression Hyperparameters

1. **Model Setup:** A Logistic Regression classifier is wrapped in scikit-learn's OneVsRestClassifier to handle multiclass classification:

```
base_classifier = LogisticRegression(random_state=42)
ovr_classifier = OneVsRestClassifier(base_classifier)
```

2. **Data Preparation:** The target labels are binarized using label_binarize to create a binary matrix representation where each column represents one class.

3. **Model Evaluation:** The evaluation includes multiple metrics:

   - F1 scores for each class
   - Macro-averaged F1 score across all classes

```
Test Set Performance:
Accuracy: 0.6601229426928416

Classification Report:
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:486: UserWarning:
  warnings.warn(
              precision    recall  f1-score   support

           0       0.54      0.96      0.69      3906
           1       0.97      0.47      0.63      2356
           2       0.92      0.55      0.69      1730
           3       0.90      0.41      0.56      2094

    accuracy                           0.66     10086
   macro avg       0.83      0.60      0.64     10086
weighted avg       0.78      0.66      0.65     10086
```

Figure 19: Test Accuracy on Logistic Regression

```
Test Set Performance:
Accuracy: 0.9460638508824113

Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.95      0.95      3906
           1       0.94      0.95      0.94      2356
           2       0.96      0.96      0.96      1730
           3       0.94      0.93      0.94      2094

    accuracy                           0.95     10086
   macro avg       0.95      0.95      0.95     10086
weighted avg       0.95      0.95      0.95     10086
```

Figure 20: Test Accuracy on Decision Tree

- ROC curves and AUC scores
- Precision-Recall curves and Average Precision scores

1.8.2 Visualization

The analysis includes two types of visualizations for each class:

- **ROC Curves:** Plot the True Positive Rate against the False Positive Rate, with the Area Under the Curve (AUC) score displayed for each class.
- **Precision-Recall Curves:** Display the trade-off between precision and recall, with the Average Precision (AP) score shown for each class.

1.8.3 Performance Metrics

The code generates comprehensive performance metrics including:

- Individual F1 scores for each class
- Macro-averaged F1 score
- Class-specific Average Precision scores
- Class-specific AUC-ROC scores

These metrics provide a detailed view of the model's performance across different classes, helping identify any class-specific strengths or weaknesses in the classification.

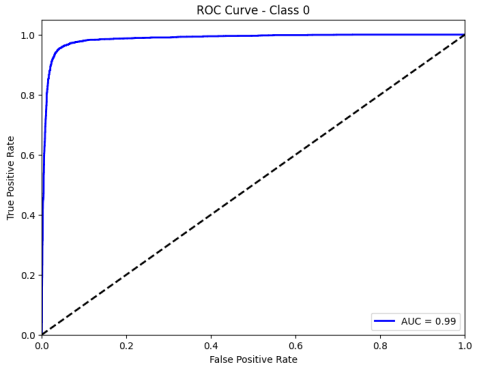

Figure 21: Household ROC Curve

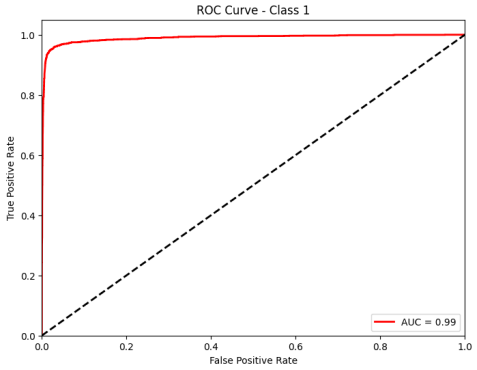

Figure 22: Books ROC Curve

1.8.4 Implementation Notes

The code uses a consistent color scheme across visualizations:

- Blue for class 0
- Red for class 1
- Green for class 2
- Purple for class 3 (if applicable)

All random operations are seeded (random_state=42) to ensure reproducibility of results.

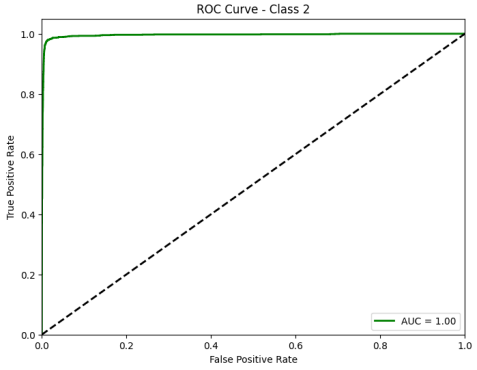

Figure 23: Clothing and Accessories ROC Curve

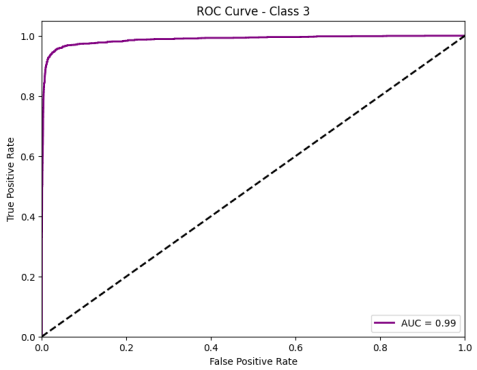

Figure 24: Electronics ROC Curve

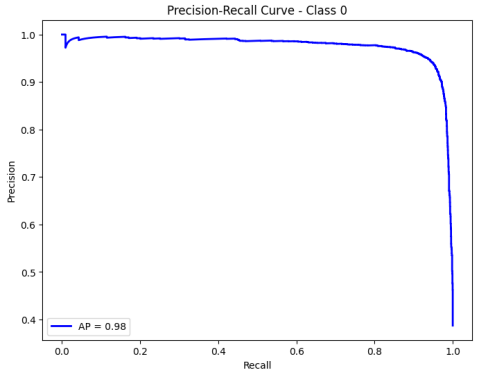

Figure 25: Household PR-Curve

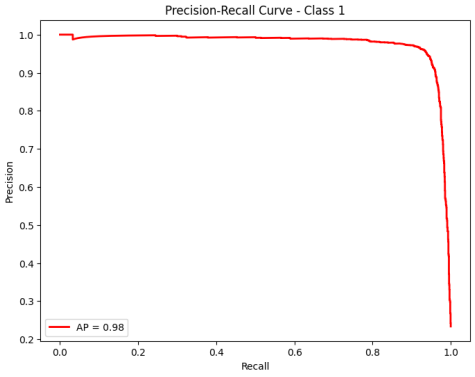

Figure 26: Books PR-Curve

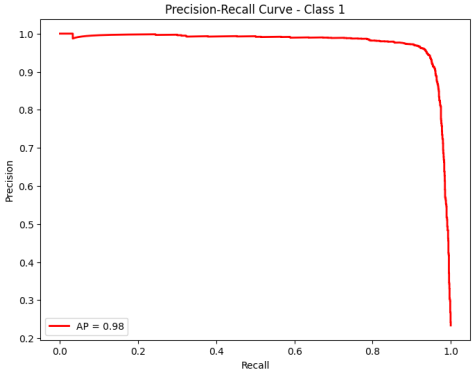

Figure 27: Clothing and Accessories PR-Curve

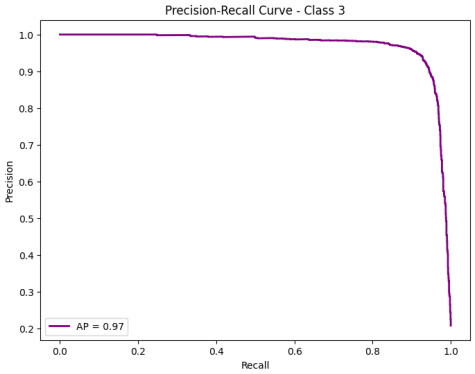

Figure 28: Electronics PR-Curve

1.9 Conclusion

Through extensive experimentation with feature engineering, model selection, and hyperparameter tuning, we have derived several key insights:

- **Feature Engineering:** The combination of all features (Description Length, Sentiment, BoW, and TF-IDF) proved most effective, yielding the highest accuracy and F1 scores across models. This suggests that capturing multiple aspects of text data (structural, semantic, and emotional) enhances classification performance.
- **Model Performance:** Among the three models tested:
  - Support Vector Machine (SVM) demonstrated superior performance in most scenarios, particularly with the first feature engineering combination
  - Decision Trees showed competitive performance, especially in the third feature engineering setup
  - Logistic Regression maintained consistent but slightly lower performance metrics
- **Training Efficiency:** SVM showed varying training times across different feature combinations, with the second feature engineering approach being notably faster (2.75s) compared to the first (41.00s).
- **OneVsRest Analysis:** The OneVsRest implementation with Logistic Regression revealed class-specific performance variations through ROC and Precision-Recall curves, providing deeper insights into the model's classification capabilities for each product category.

These findings suggest that while model selection is important, the choice and combination of features play a crucial role in classification performance. The trade-off between computational efficiency and model accuracy should be considered based on specific use-case requirements.