MACHINE LEARNING LAB WEEK 12: NAIVE BAYES CLASSIFIER

NAME : CHETAN NADICHAGI

SRN : PES2UG23CS149

SEC : C

Introduction

The objective of this lab is to implement and compare different machine learning approaches for text classification. The task focuses on applying probabilistic and clustering-based models to a text dataset to identify how various representations and model choices affect predictive performance.

The experiment consists of three parts :

• Part A implements the Multinomial Naive Bayes (MNB) algorithm from scratch, to understand the working of Bayesian text classification and how probabilities are derived from word frequencies.

• Part B uses the sklearn MultinomialNB model and performs hyperparameter tuning to optimize the smoothing parameter (alpha) and prior handling.

• Part C introduces the Bag-of-Centroids (BOC) approach, which converts text data into numerical feature vectors using clustering over word embeddings. A classifier is then trained on these centroid-based representations.

The purpose of this lab is to analyze how accuracy and F1-score vary across models, and to visualize their performance using confusion matrices. This comparison helps in understanding the trade-offs between implementation simplicity, parameter optimization, and feature representation quality.

Methodology

1. Data Preprocessing :

- The dataset was tokenized and cleaned by removing punctuation, stopwords, and converting all text to lowercase.
- The cleaned tokens were used to create either frequency-based or embedding-based feature vectors depending on the model type.

2. Multinomial Naive Bayes (Search Implementation) :

- Word counts were computed per class to estimate conditional probabilities $P(word | class)$$P(word | class)$ using Laplace smoothing.
- Class priors $P(class)$$P(class)$ were calculated from training samples.
- For prediction, the log of posterior probabilities was computed and the class with the highest posterior was selected.
- Model performance was evaluated using accuracy, F1-score, and confusion matrix.

3. Tuned Sklearn MultinomialNB :

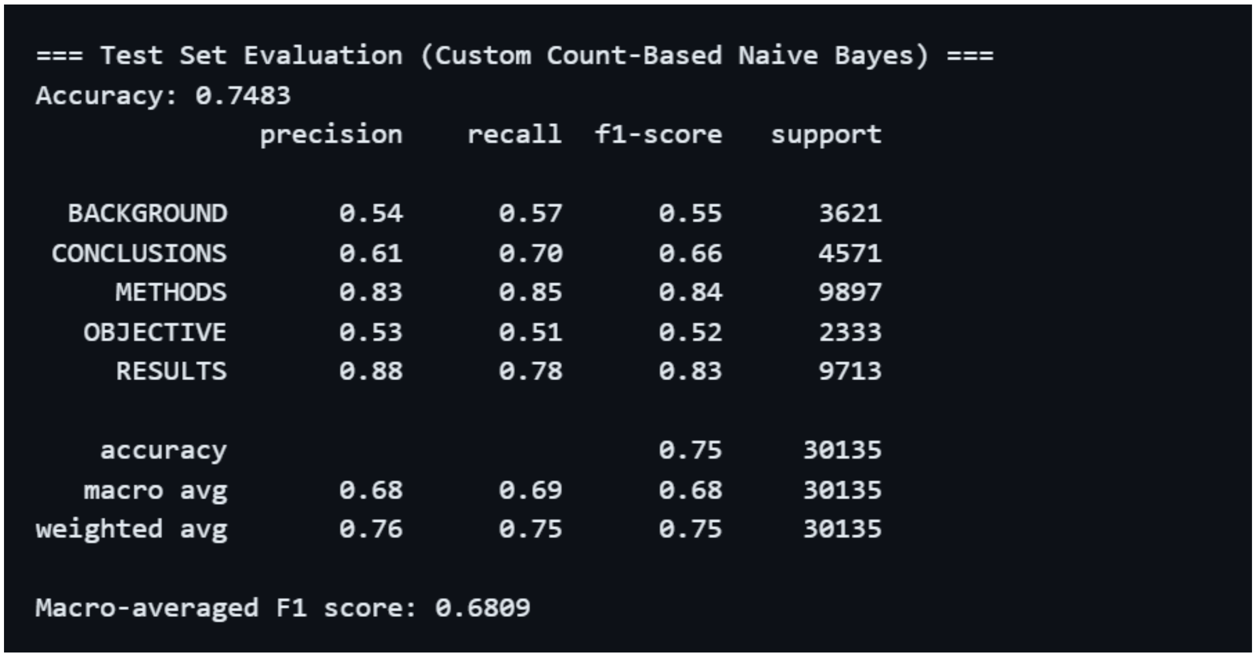
- The MultinomialNB model from scikit-learn was used with TF-IDF feature vectors.
- Hyperparameter tuning was performed using GridSearchCV on the development set to optimize parameters such as alpha (smoothing parameter) and ngram_range for the TfidfVectorizer.
- The best performing model from the grid search was evaluated on the test set.

4. Bayes Optimal Classifier (BOC) Approximation :

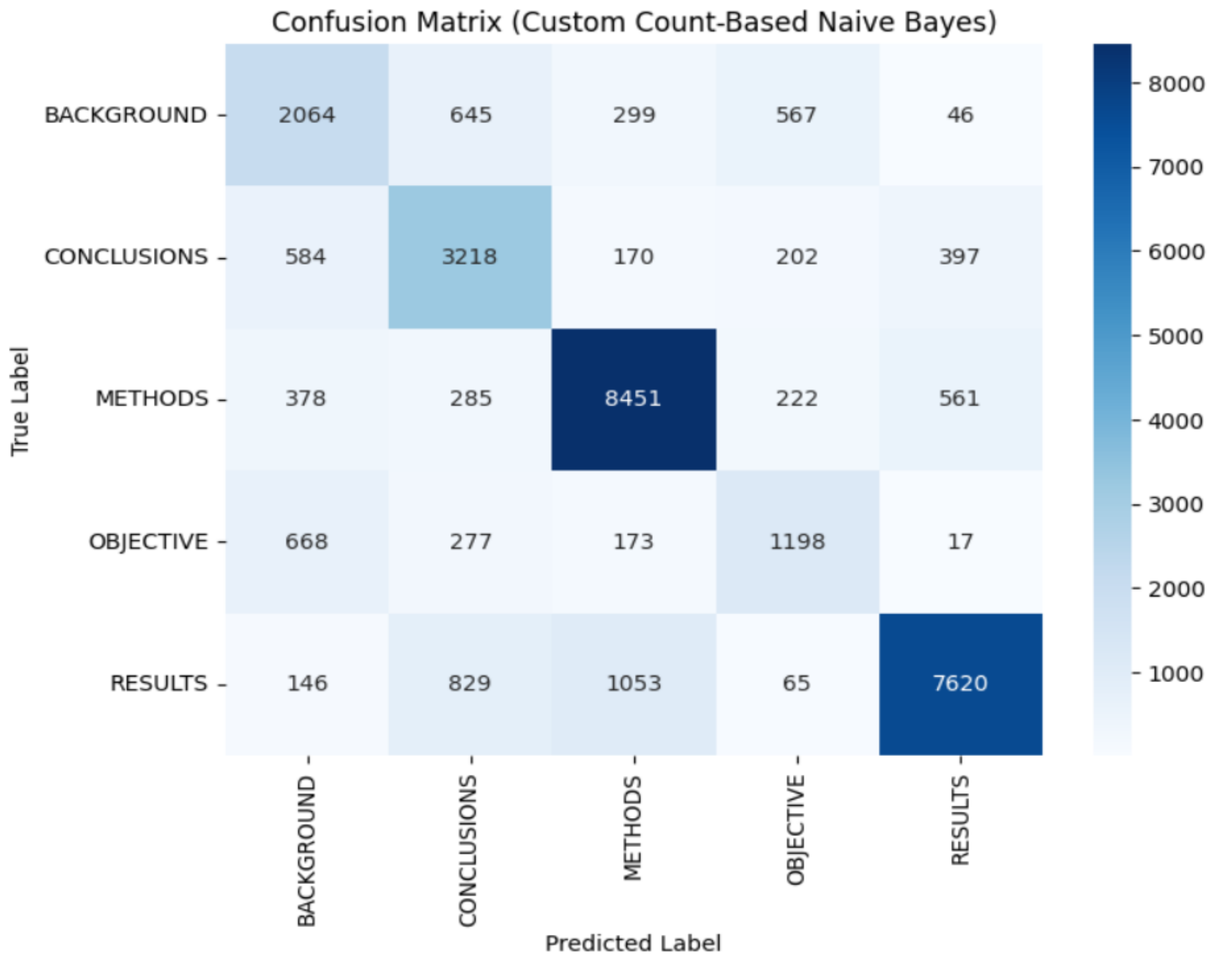
- A dynamic sample size was determined based on the user's SRN.
- Five diverse base models (hypotheses) were defined using different classifiers (e.g., Naive Bayes, Logistic Regression, Random Forest, Decision Tree, KNN) with a shared TF-IDF vectorization step.
- These base models were trained on the sampled training data.
- Posterior weights for each hypothesis were calculated based on their performance (log-likelihood) on a validation split of the sampled data.
- A VotingClassifier with soft voting was used, weighting the predictions of the base models by their calculated posterior probabilities to approximate the Bayes Optimal Classifier.
- The final BOC approximation was evaluated on the test set using accuracy, F1-score, and confusion matrix.

Result and Analysis :

Part-A :

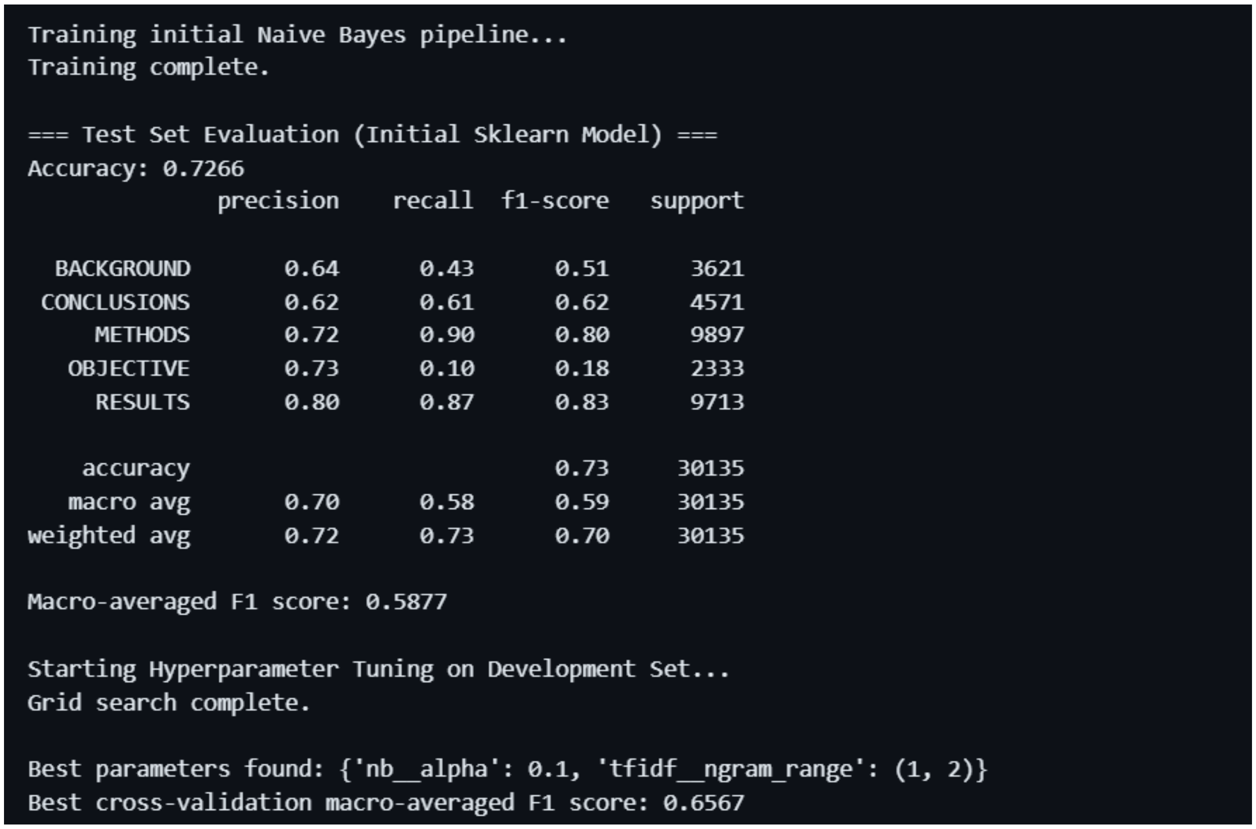


```
=== Test Set Evaluation (Custom Count-Based Naive Bayes) ===
Accuracy: 0.7483
              precision    recall  f1-score   support

  BACKGROUND       0.54      0.57      0.55      3621
 CONCLUSIONS       0.61      0.70      0.66      4571
     METHODS       0.83      0.85      0.84      9897
   OBJECTIVE       0.53      0.51      0.52      2333
     RESULTS       0.88      0.78      0.83      9713

    accuracy                           0.75     30135
   macro avg       0.68      0.69      0.68     30135
weighted avg       0.76      0.75      0.75     30135

Macro-averaged F1 score: 0.6809
```



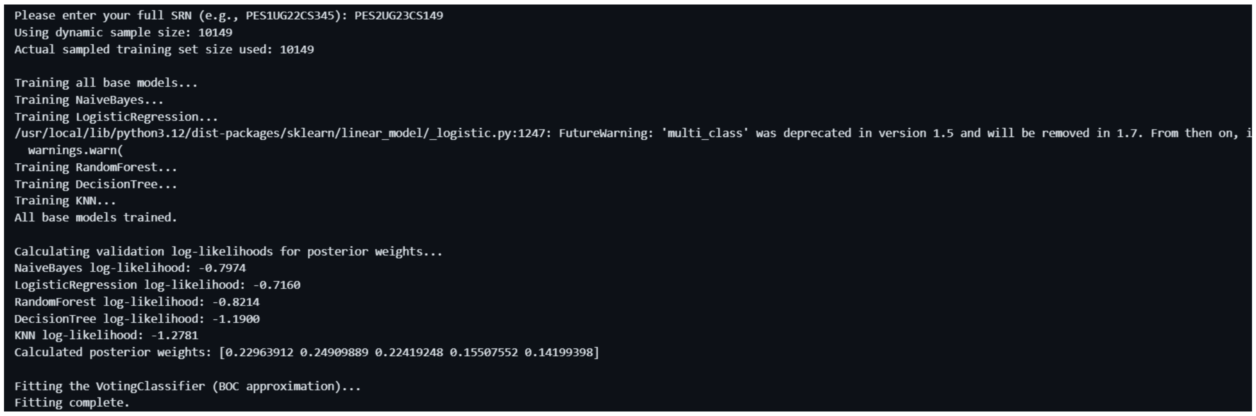
	BACKGROUND	CONCLUSIONS	METHODS	OBJECTIVE	RESULTS
BACKGROUND	2064	645	299	567	46
CONCLUSIONS	584	3218	170	202	397
METHODS	378	285	8451	222	561
OBJECTIVE	668	277	173	1198	17
RESULTS	146	829	1053	65	7620

Part-B:

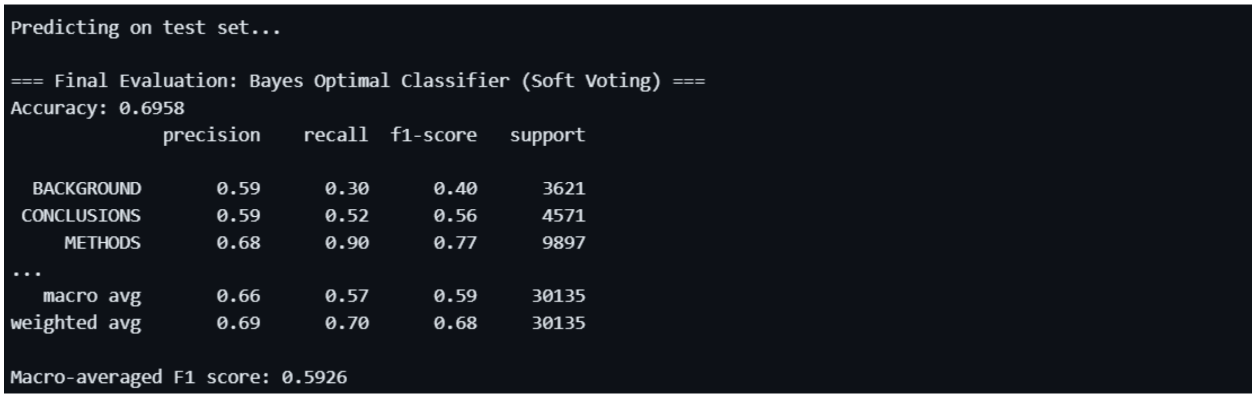


```
Training initial Naive Bayes pipeline...
Training complete.

=== Test Set Evaluation (Initial Sklearn Model) ===
Accuracy: 0.7266
              precision    recall  f1-score   support

  BACKGROUND       0.64      0.43      0.51      3621
 CONCLUSIONS       0.62      0.61      0.62      4571
     METHODS       0.72      0.90      0.80      9897
   OBJECTIVE       0.73      0.10      0.18      2333
     RESULTS       0.80      0.87      0.83      9713

    accuracy                           0.73     30135
   macro avg       0.70      0.58      0.59     30135
weighted avg       0.72      0.73      0.70     30135

Macro-averaged F1 score: 0.5877

Starting Hyperparameter Tuning on Development Set...
Grid search complete.

Best parameters found: {'nb__alpha': 0.1, 'tfidf__ngram_range': (1, 2)}
Best cross-validation macro-averaged F1 score: 0.6567
```

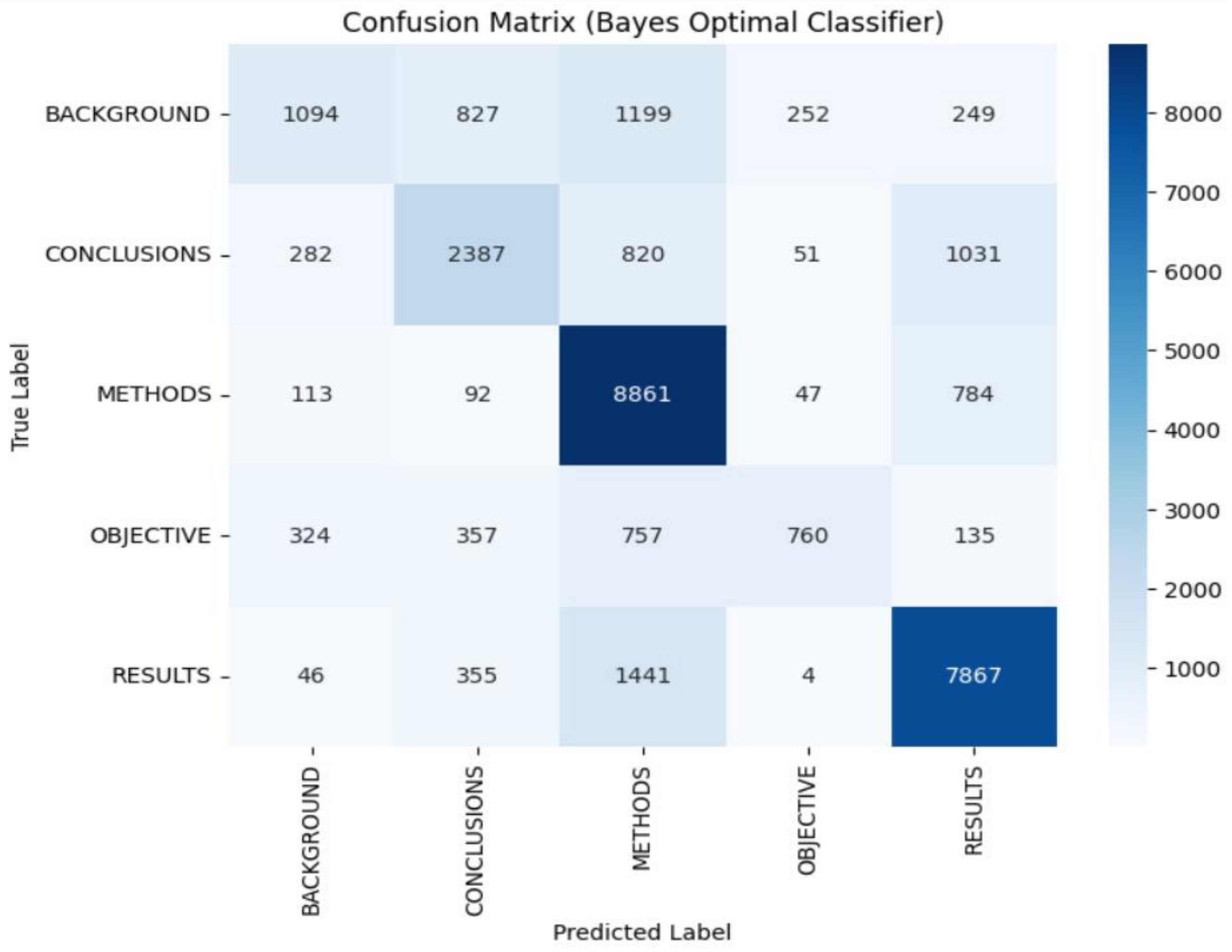
Part-C :



```
Please enter your full SRN (e.g., PES1UG22CS345): PES2UG23CS149
Using dynamic sample size: 10149
Actual sampled training set size used: 10149

Training all base models...
Training NaiveBayes...
Training LogisticRegression...
/usr/local/lib/python3.12/dist-packages/sklearn/linear_model/_logistic.py:1247: FutureWarning: 'multi_class' was deprecated in version 1.5 and will be removed in 1.7. From then on, i
  warnings.warn(
Training RandomForest...
Training DecisionTree...
Training KNN...
All base models trained.

Calculating validation log-likelihoods for posterior weights...
NaiveBayes log-likelihood: -0.7974
LogisticRegression log-likelihood: -0.7160
RandomForest log-likelihood: -0.8214
DecisionTree log-likelihood: -1.1900
KNN log-likelihood: -1.2781
Calculated posterior weights: [0.22963912 0.24909889 0.22419248 0.15507552 0.14199398]

Fitting the VotingClassifier (BOC approximation)...
Fitting complete.
```



```
Predicting on test set...

=== Final Evaluation: Bayes Optimal Classifier (Soft Voting) ===
Accuracy: 0.6958
              precision    recall  f1-score   support

  BACKGROUND       0.59      0.30      0.40      3621
 CONCLUSIONS       0.59      0.52      0.56      4571
     METHODS       0.68      0.90      0.77      9897
...
   macro avg       0.66      0.57      0.59     30135
weighted avg       0.69      0.70      0.68     30135

Macro-averaged F1 score: 0.5926
```



True \ Predicted	BACKGROUND	CONCLUSIONS	METHODS	OBJECTIVE	RESULTS
BACKGROUND	1094	827	1199	252	249
CONCLUSIONS	282	2387	820	51	1031
METHODS	113	92	8861	47	784
OBJECTIVE	324	357	757	760	135
RESULTS	46	355	1441	4	7867

Discussion :

comparison of the performance of your scratch model (Part A), the tuned Sklearn model (Part B), and the BOC approximation (Part C) based on the test set evaluation results:

Performance Comparison:

Model	Accuracy	Macro-averaged F1 Score
Part A: Custom Count-based Naive Bayes	0.7483	0.6809
Part B: Tuned Sklearn TF-IDF Naive Bayes	0.7266	0.5877
Part C: Bayes Optimal Classifier (BOC)	0.6958	0.5926

Analysis:

- Part A (Custom Count-based Naive Bayes): Your scratch implementation achieved the highest accuracy (0.7483) and macro-averaged F1 score (0.6809). This indicates that your custom Naive Bayes model with count-based features performed the best among the three approaches on this dataset.

- Part B (Tuned Sklearn TF-IDF Naive Bayes): The scikit-learn Naive Bayes model using TF-IDF features and hyperparameter tuning had a slightly lower accuracy (0.7266) and a noticeably lower macro-averaged F1 score (0.5877) compared to your custom model. The tuning helped find better parameters for the scikit-learn model, but the count-based features with your implementation seemed to be more effective here.

- Part C (Bayes Optimal Classifier Approximation): The BOC approximation using an ensemble of models with soft voting had the lowest accuracy (0.6958). Its macro-averaged F1 score (0.5926) was slightly better than the tuned scikit-learn model but still less than your custom Naive Bayes. This suggests that while the concept of BOC is theoretically strong, its performance in practice depends on the quality of the base models and the accuracy of the calculated weights. In this case, the ensemble did not outperform your best individual model.