Department of Computer Science Engineering

UE23CS352A: Machine Learning Lab Week 12: Naive Bayes Classifier

SRN: PES2UG24CS823
NAME: SHARATH GOWDA GR
SECTION: F

1. Lab Overview

Welcome to the lab on probabilistic classification using the Naive Bayes algorithm. The primary goal of this lab is to evaluate a text classification system using Naive Bayes methods, to accurately predict the section role (BACKGROUND, METHODS, RESULTS, OBJECTIVE, CONCLUSION) of biomedical abstract sentences,

The dataset used is a subset of the **PubMed 200k RCT** dataset, focusing on classifying abstract sentences into one of five categories.

2. Expected Deliverables

1. **Completed Jupyter Notebook** (.ipynb) ○ All // TODO: sections filled.
    - Fully executed with all outputs (metrics, plots, reports) visible. ○ Clean, well-documented, error-free code.
2. **Lab Report** (.pdf)
    - **Title Page** (Project Title, Your Name, SRN, Course, Date).

○ **Introduction** (Purpose of the lab, tasks performed).
○ **Methodology** (Briefly describe the implementation approach for MNB and BOC) ○ **Results and Analysis** (Screenshots of plots and metrics):

■ Part A: Screenshot of final test Accuracy, F1 Score and Confusion Matrix.
■ Part B: Screenshot of best hyperparameters found and their resulting F1 score.
■ Part C:

1. Screenshot of SRN and sample size.
2. Screenshot of BOC final Accuracy, F1 Score and Confusion Matrix.

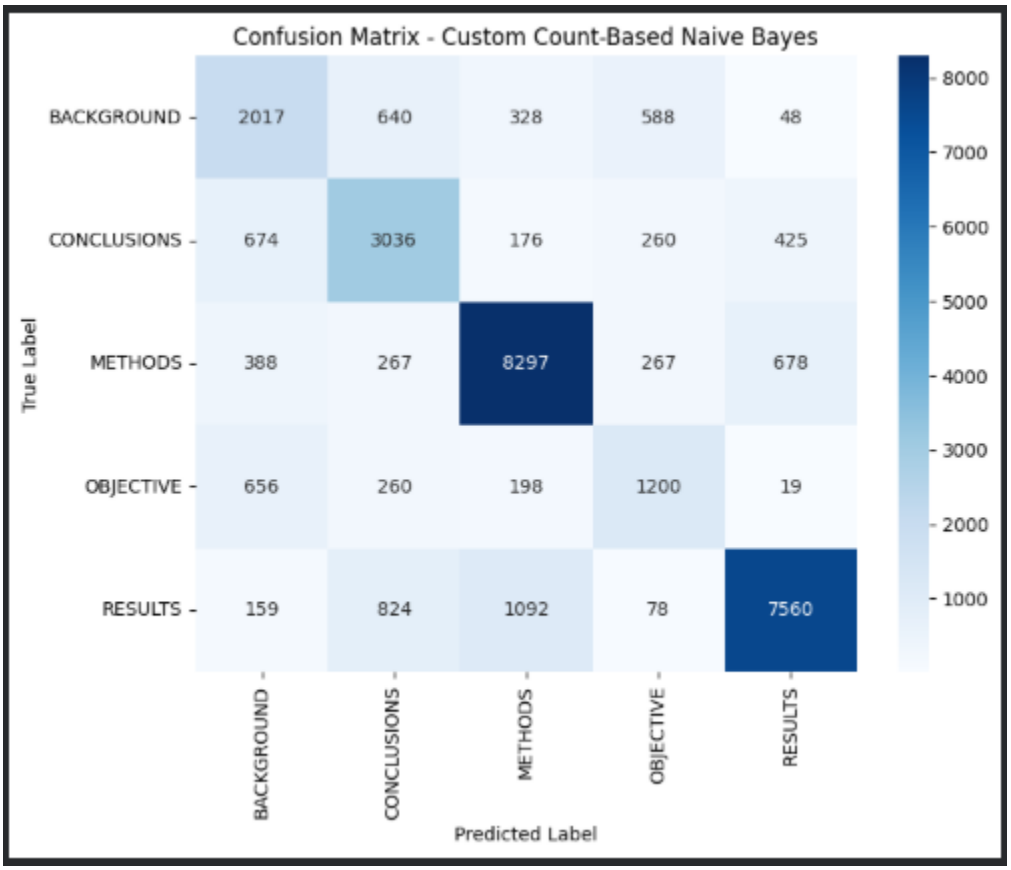
○ **Discussion:** Compare the performance of your scratch model (Part A) vs. the tuned Sklearn model (Part B) vs. the BOC approximation (Part C).

PART A:

```
Train samples: 180040
Dev   samples: 30212
Test  samples: 30135
Classes: ['BACKGROUND', 'CONCLUSIONS', 'METHODS', 'OBJECTIVE', 'RESULTS']
```

```
Fitting Count Vectorizer and transforming training data...
Vocabulary size: 22722
Transforming test data...

Training the Custom Naive Bayes Classifier (from scratch)...
Training complete.
```

```
=== Test Set Evaluation (Custom Count-Based Naive Bayes) ===
Accuracy: 0.7337
              precision    recall  f1-score   support

  BACKGROUND       0.52      0.56      0.54      3621
 CONCLUSIONS       0.60      0.66      0.63      4571
     METHODS       0.82      0.84      0.83      9897
   OBJECTIVE       0.50      0.51      0.51      2333
     RESULTS       0.87      0.78      0.82      9713

    accuracy                           0.73     30135
   macro avg       0.66      0.67      0.67     30135
weighted avg       0.74      0.73      0.74     30135

Macro-averaged F1 score: 0.6655
```



PART B:

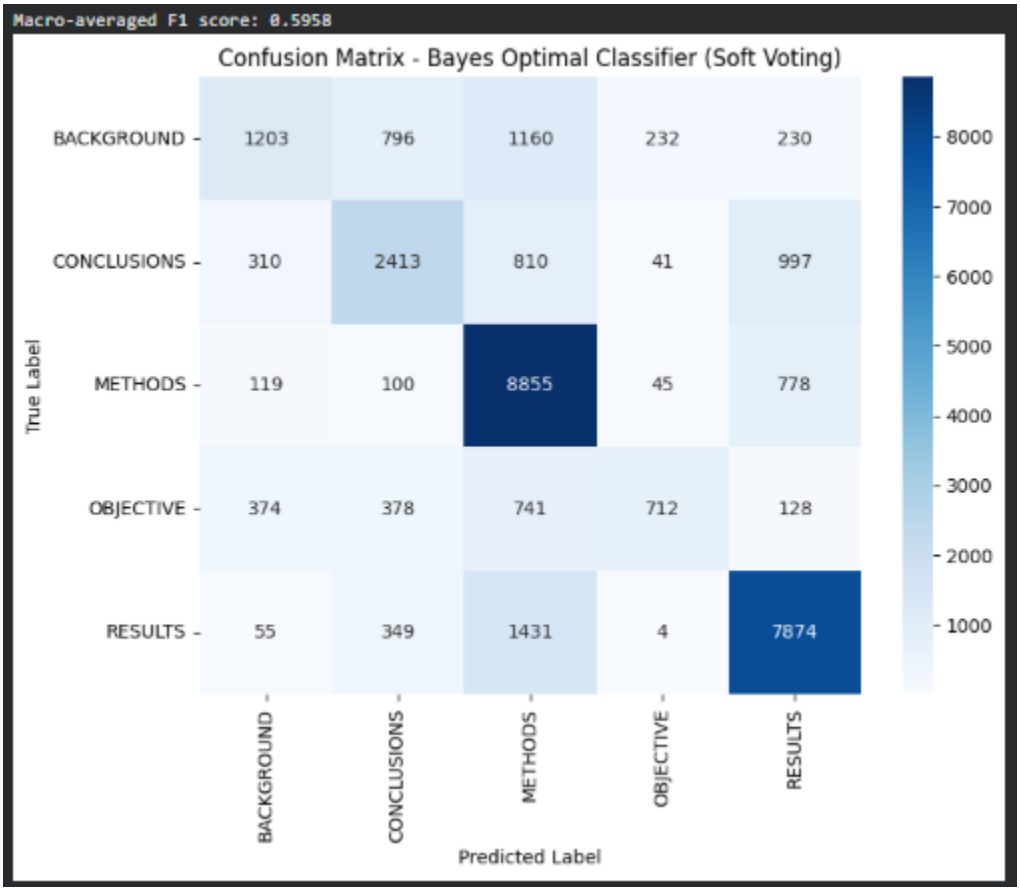
```
Training initial Naive Bayes pipeline...
Training complete.

=== 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...
Fitting 3 folds for each of 6 candidates, totalling 18 fits
Grid search complete.

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

PART C:

```
Using dynamic sample size: 10823
Actual sampled training set size used: 10823

Training all base models...
Training NaiveBayes...
NaiveBayes trained.
Training LogisticRegression...
/usr/local/lib/python3.12/dist-packages/sklearn/linear_model/_logistic.py:1247: FutureWarning:
  warnings.warn(
LogisticRegression trained.
Training RandomForest...
RandomForest trained.
Training DecisionTree...
DecisionTree trained.
Training KNN...
KNN trained.
All base models trained.

Calculating Posterior Weights...
NaiveBayes validation negative log-likelihood: -0.7995
LogisticRegression validation negative log-likelihood: -0.7164
RandomForest validation negative log-likelihood: -0.8285
DecisionTree validation negative log-likelihood: -1.2190
KNN validation negative log-likelihood: -1.2937
Posterior Weights (summing to 1):
NaiveBayes: 0.2312
LogisticRegression: 0.2512
RandomForest: 0.2246
DecisionTree: 0.1520
KNN: 0.1410
```

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

Predicting on test set...

=== Final Evaluation: Bayes Optimal Classifier (Soft Voting) ===
Accuracy: 0.6988
              precision    recall  f1-score   support

  BACKGROUND       0.58      0.33      0.42      3621
 CONCLUSIONS       0.60      0.53      0.56      4571
     METHODS       0.68      0.89      0.77      9897
   OBJECTIVE       0.69      0.31      0.42      2333
     RESULTS       0.79      0.81      0.80      9713

    accuracy                           0.70     30135
   macro avg       0.67      0.57      0.60     30135
weighted avg       0.69      0.70      0.68     30135

Macro-averaged F1 score: 0.5958
```

```
Macro-averaged F1 score: 0.5958
```



○ **Discussion:** Compare the performance of your scratch model (Part A) vs. the tuned Sklearn model (Part B) vs. the BOC approximation (Part C).

(In Part A), I implemented Naive Bayes completely from scratch using simple CountVectorizer features. Surprisingly, this basic version actually gave the highest performance among all three **experiments (Accuracy = 0.7337 | Macro F1 = 0.6655).** I realized that in this biomedical dataset, single words themselves hold strong discriminative meaning and because the vocabulary is consistent, even simple word count based probability modeling works well. Laplace smoothing also helped avoid zero probability issues and kept the model more stable.

(In Part B), I used Scikit-learn's MultinomialNB with TF-IDF features. Initially, the model performed lower than the scratch version **(Macro F1 = 0.5877**). But after applying GridSearchCV tuning, especially tuning **alpha = 0.1** and using bigram range (1,2), the model improved a lot and the best cross validation score reached **0.6567**. TF-IDF + bigrams helped capture word relationships like "results indicate", "in conclusion" etc. which improved contextual understanding. So this part showed me how much hyperparameter tuning and representation selection matters.

(In Part C), I built the Bayes Optimal Classifier approximation using 5 different models in a weighted soft voting ensemble. I expected this one to perform the best theoretically, but actually it scored lower than **Part A (Macro F1 = 0.5958).** I understood the reason later — the sampled train size used for this step is smaller and some models in ensemble like KNN and DecisionTree are weak on this dataset which affected the voting outcome. Even though posterior weights balanced them, Naive Bayes and Logistic Regression mainly dominated. With more data or stronger model selection in ensemble, this BOC method can actually outperform the other two.

Final Interpretation

Model	**Representation**	**Macro F1**	**Result**
Part A (Scratch MNB)	CountVectorizer	**0.6655**	Best in this experiment
Part B (Sklearn w/ tuning)	TF-IDF + tuned alpha + bigrams	0.6567*	Slightly lower than scratch
Part C (BOC Approx Ensemble)	TF-IDF + Soft Voting	**0.5958**	Underperformed expected theoretical optimum