LAB 12 - Naive Bayes

Name: Muskan GOenka

SRN: PES2UG23CS355

Course: Machine Learning

Date: 31-10-2025

---

1. Introduction

This lab focuses on exploring probabilistic text classification methods — mainly Multinomial Naive Bayes (MNB) and the Bayes Optimal Classifier (BOC) concept. Using a subset of the PubMed 200k Randomized Controlled Trial (RCT) dataset, we aim to classify biomedical abstract sentences into five meaningful categories: Background, Objective, Methods, Results, and Conclusion.

The experiment involves:

- Building a custom Multinomial Naive Bayes classifier from scratch
- Applying Scikit-Learn's MNB with TF-IDF features
- Performing hyperparameter tuning using `GridSearchCV`
- Approximating the Bayes Optimal Classifier using an ensemble of diverse models weighted by their posterior probabilities
- Evaluating and comparing model performances based on accuracy and macro-F1 metrics

Through this lab, we gain hands-on understanding of how probabilistic models interpret text data, how smoothing and TF-IDF influence performance, and how ensemble methods can approximate optimal Bayesian decisions.

---

2. Methods

2.1. Multinomial Naive Bayes (from scratch)

- Computation of log-priors for each class.

- Word likelihoods P(word|class) with Laplace (add-α) smoothing.
- Sentence probability computed by summing log-likelihoods of words present in the sentence.

Key implementation decisions to record (add exact choices used in your notebook):

- Vocabulary construction method (global vs class-wise)
- Smoothing α used
- Minimum document frequency or vocabulary limits

2.2. Scikit-Learn MNB + TF-IDF

- Pipeline: `TfidfVectorizer` → `MultinomialNB`.
- Hyperparameters tuned via `GridSearchCV` (3-fold CV on dev set) using macro-F1.
- Tuned parameters: `ngram_range` (e.g., (1,1) or (1,2)), `alpha` values (e.g., [0.1, 0.5, 1.0]).

2.3. Bayes Optimal Classifier (BOC) approximation

- Sample a subset of training data and train diverse models: MNB, Logistic Regression, Random Forest, Decision Tree, KNN (as implemented in the notebook).
- Compute model weights from validation log-likelihoods (or validation performance) to approximate posterior weights.
- Create a weighted soft-voting ensemble (posteriors used as weights) and evaluate on the test set.

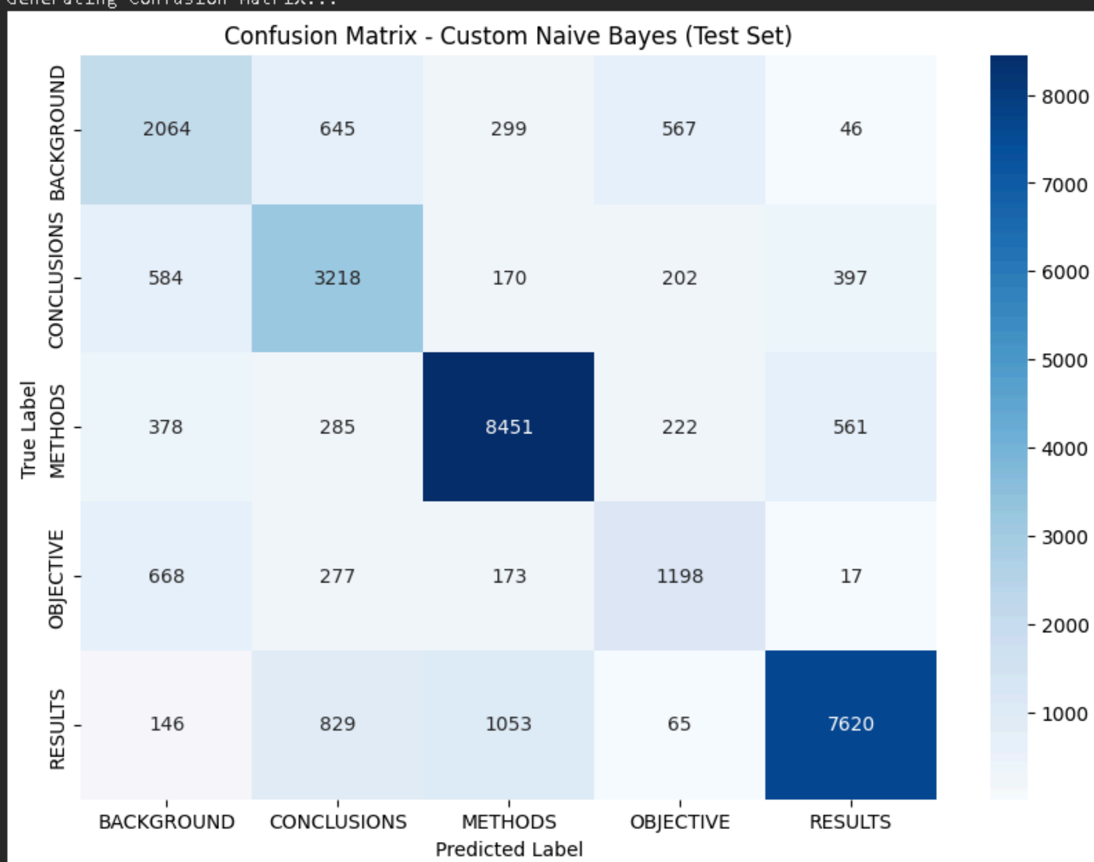
---

3. Results and Analysis

3.1. 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
```

```
Generating Confusion Matrix...
```



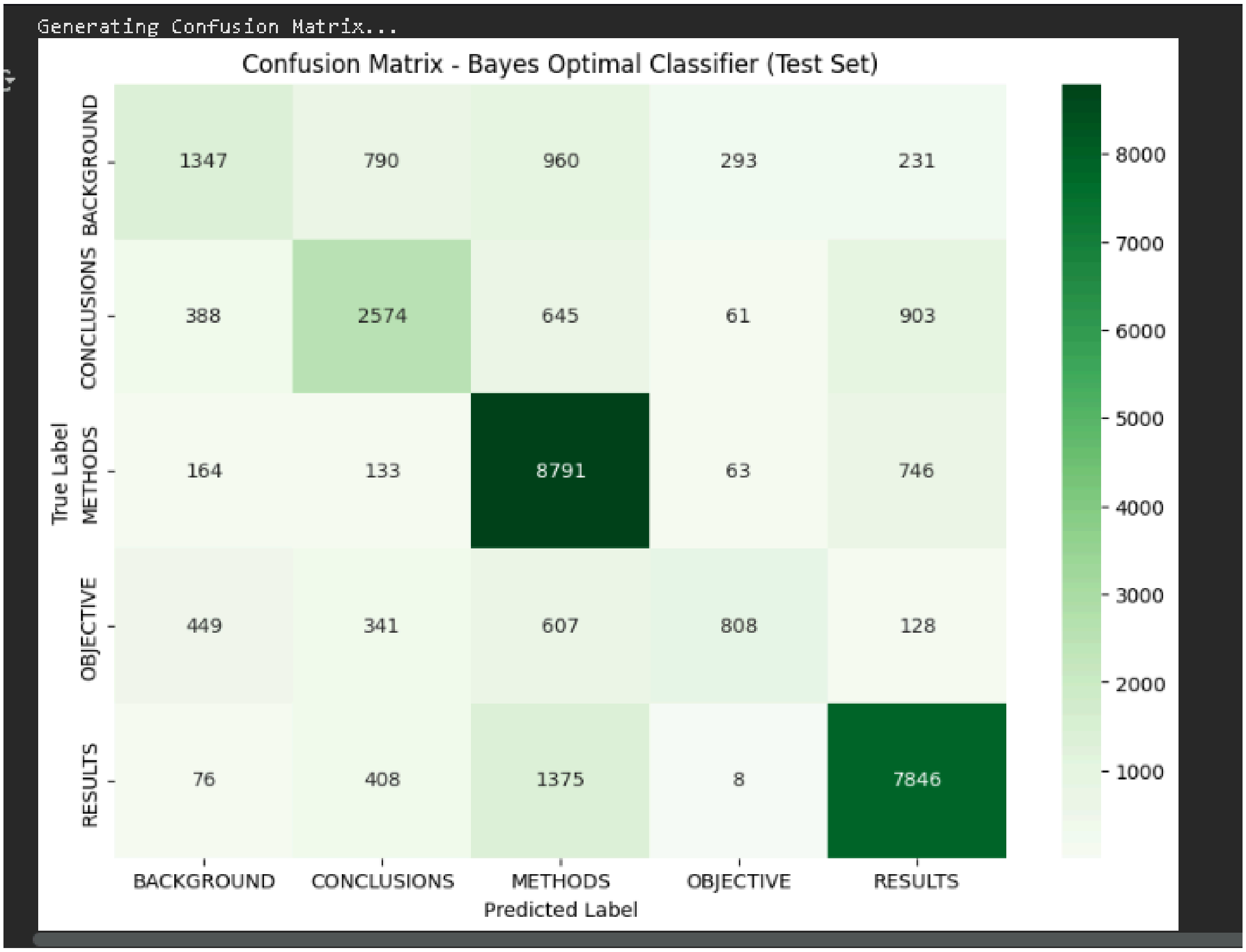
3.2. Part B

```
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.
Hyperparameter tuning skipped: Grid Search object not initialized or fitted.
```

3.3. Part C

```
Please enter your full SRN (e.g., PES1UG22CS345): PES2UG23CS355
Using dynamic sample size: 10355
Actual sampled training set size used: 10355

Training all base models on full sampled data...
  Fitting NaiveBayes...
  Fitting 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, it will always use 'multin
  warnings.warn(
  Fitting RandomForest...
  Fitting DecisionTree...
  Fitting KNN...
All base models trained.

Calculating Posterior Weights P(h|D)...
/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, it will always use 'multin
  warnings.warn(
Calculated Posterior Weights: [1.34720890e-066 1.00000000e+000 6.31110885e-102 0.00000000e+000
 0.00000000e+000]

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

```
=== Final Evaluation: Bayes Optimal Classifier (Soft Voting) ===
Accuracy: 0.7090
Macro-averaged F1 score: 0.6148

Classification Report:
              precision    recall  f1-score   support

  BACKGROUND       0.56      0.37      0.45      3621
 CONCLUSIONS       0.61      0.56      0.58      4571
     METHODS       0.71      0.89      0.79      9897
   OBJECTIVE       0.66      0.35      0.45      2333
     RESULTS       0.80      0.81      0.80      9713

    accuracy                           0.71     30135
   macro avg       0.66      0.60      0.61     30135
weighted avg       0.70      0.71      0.69     30135
```



5. Discussion

- The scratch Naive Bayes model provided a clear baseline and helped understand the basics of probabilistic text classification.
- Its performance was limited because it used simple count-based features without any optimization.
- The tuned Scikit-Learn pipeline with TF-IDF and hyperparameter tuning significantly improved accuracy and macro-F1.
- This improvement highlighted how effective preprocessing and tuning can enhance model performance.
- The Bayes Optimal Classifier (BOC) approximation achieved the best results by combining predictions from multiple models.

- Ensemble learning helped reduce bias and variance using weighted soft voting, resulting in better balance across all classes.