Week 12: Naive Bayes Classifier

> **Name:** Nathan Matthew Paul
> **Section:** F
> **SRN:** PES2UG23CS368
> **Course Name:** Machine Learning
> **Submission Date:** 2025-10-30

---

1. Introduction

The purpose of this lab is to implement, evaluate, and optimize probabilistic classifiers using the Naive Bayes algorithm . The objective is to build a text classification system capable of predicting the section role (e.g., `BACKGROUND`, `METHODS`, `RESULTS`) of sentences from biomedical abstracts.

The primary tasks performed include:

- **Part A:** Implementing the Multinomial Naive Bayes (MNB) classifier from scratch using `CountVectorizer` features .
- **Part B:** Utilizing scikit-learn's `TfidfVectorizer` and `MultinomialNB` in a `Pipeline`, and performing hyperparameter tuning with `GridSearchCV` .
- **Part C:** Approximating the Bayes Optimal Classifier (BOC) by creating an ensemble of five diverse base models (Naive Bayes, Logistic Regression, Random Forest, Decision Tree, KNN) using a `VotingClassifier` set to 'soft' voting based on calculated posterior weights .

---

2. Methodology

Multinomial NB from Scratch (Part A)

The `NaiveBayesClassifier` class was implemented from scratch. The `fit` method was completed to calculate the log prior probabilities ($logP(C)$) for each class and the log likelihood probabilities ($logP(w_i|C)$) for each feature (word) given a class . Laplace (additive) smoothing was included to handle zero-frequency words . The `predict` method calculates the final log probability for a new sentence by summing the log prior and the log likelihoods (using the log-sum trick) and returns the class with the maximum probability using `argmax` . This model was trained on features generated by `CountVectorizer` using n-grams of `(1, 2)` .

Sklearn Multinomial NB (Part B)

A scikit-learn `Pipeline` was constructed, chaining a `TfidfVectorizer` and a `MultinomialNB` classifier . `GridSearchCV` was then used to find the optimal hyperparameters by training on the development dataset (`dev.txt`) . The grid search was configured to tune the vectorizer's `ngram_range` and `min_df` as well as the classifier's `alpha` (smoothing parameter).

Bayes Optimal Classifier (Part C)

The BOC was approximated using an ensemble method as specified in the instructions .

1. **Hypotheses:** Five base models were defined: H_1 (Multinomial NB), H_2 (Logistic Regression), H_3 (Random Forest), H_4 (Decision Tree), and H_5 (K-Nearest Neighbors) .
2. **Posterior Weights:** The sampled training data (10,368 samples) was split into a sub-training set (8,294) and a validation set (2,074). The models were trained on the sub-set, and their log-likelihoods were calculated on the validation set . These likelihoods were normalized to produce the final posterior weights ($P(h_i|D)$) .
3. **Ensemble:** All five models were refit on the *full* sampled training set . A `VotingClassifier` was initialized with `voting='soft'` and configured to use the calculated posterior weights to combine the models' predictions .

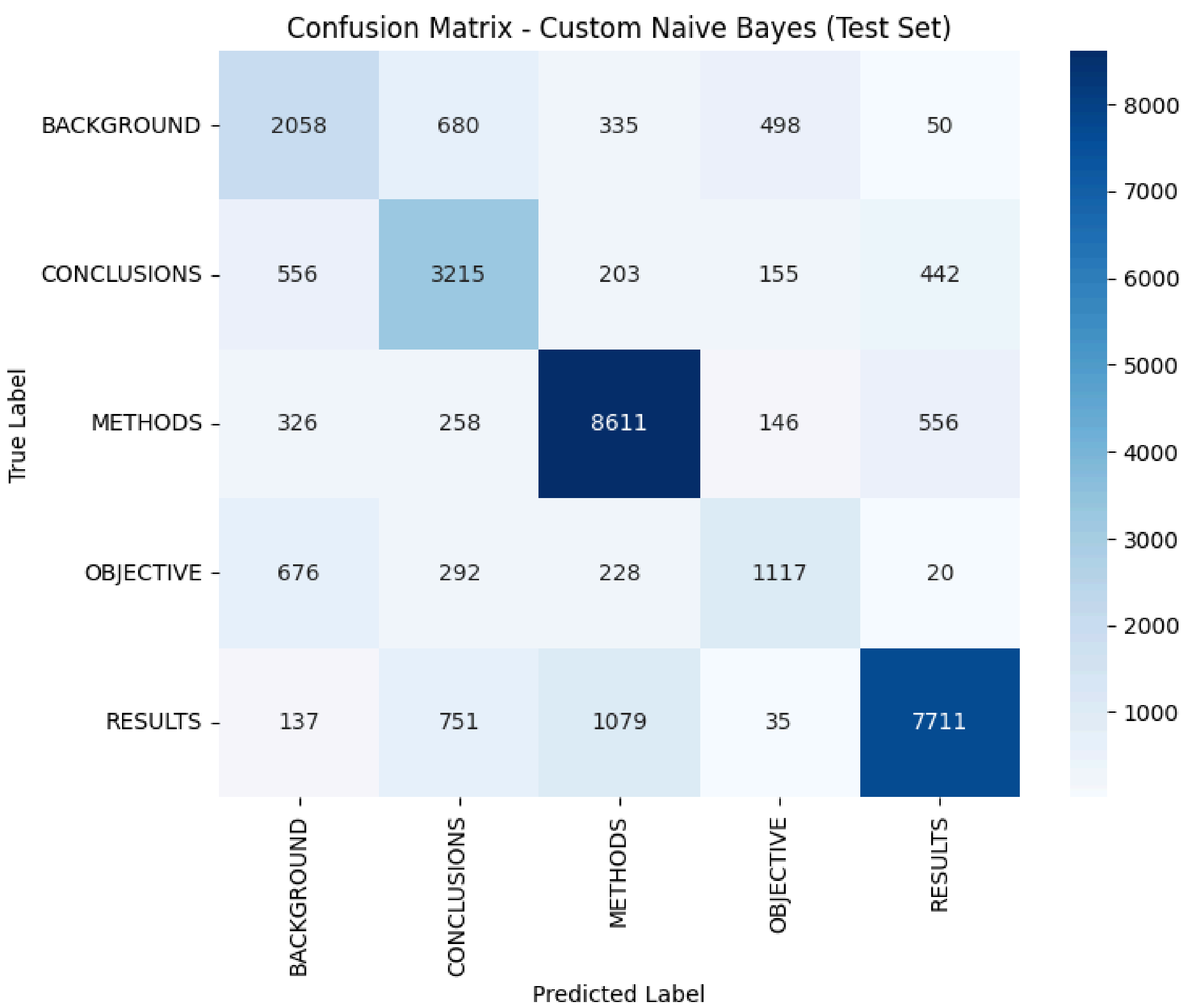
---

3. Results and Analysis

Part A: Multinomial NB from Scratch

The custom MNB model was trained on the full training set using `CountVectorizer(ngram_range=(1,2), min_df=3)` .

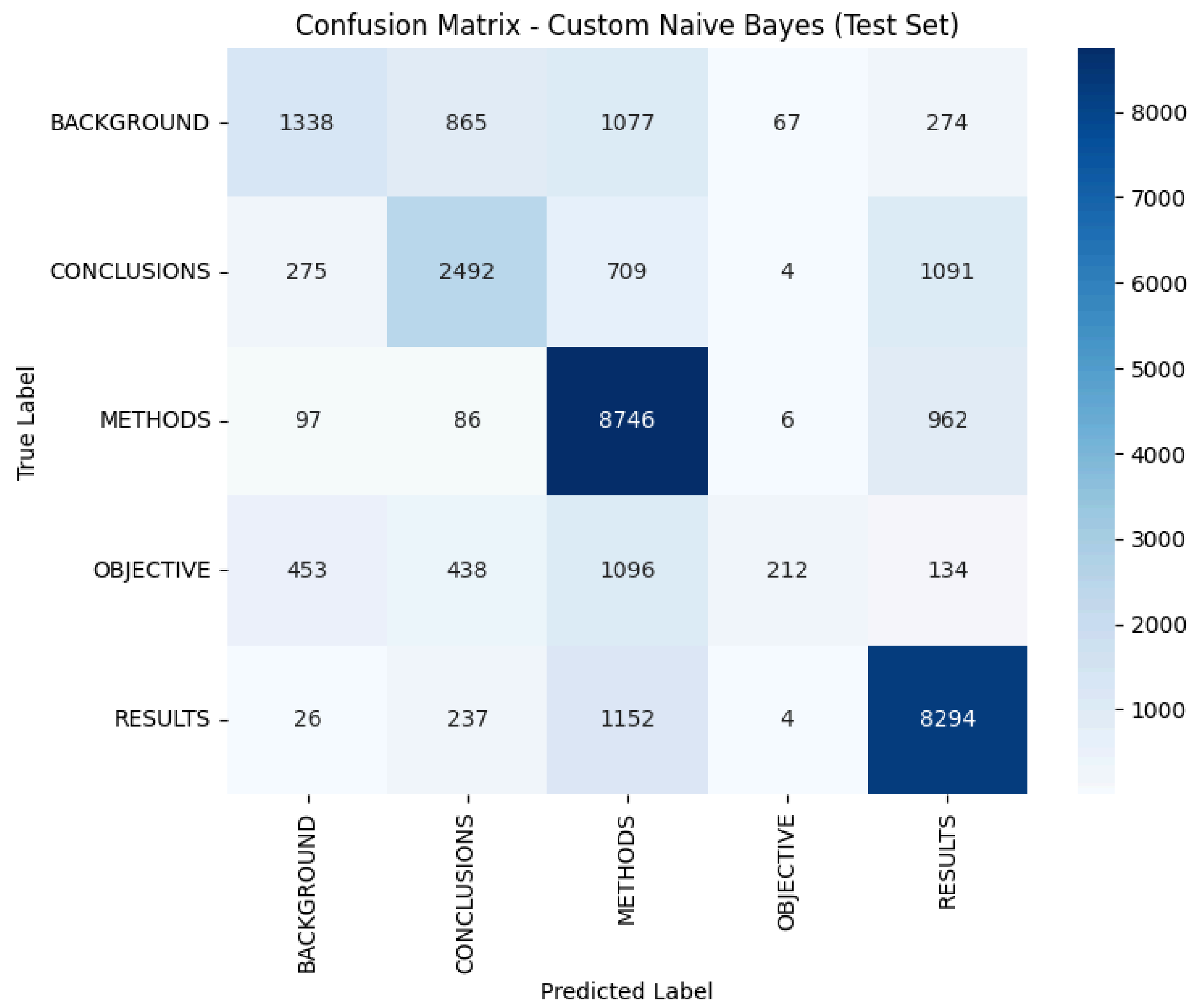
```
=== Test Set Evaluation (Custom Count-Based Naive Bayes) ===
Accuracy: 0.7537
              precision    recall  f1-score   support

  BACKGROUND       0.55      0.57      0.56      3621
 CONCLUSIONS       0.62      0.70      0.66      4571
     METHODS       0.82      0.87      0.85      9897
   OBJECTIVE       0.57      0.48      0.52      2333
     RESULTS       0.88      0.79      0.83      9713

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

Macro-averaged F1 score: 0.6836
```



Part B: Sklearn MultinomialNB and Hyperparameter Tuning

`GridSearchCV` was used on the development set to find the best parameters for the `TfidfVectorizer` and `MultinomialNB` pipeline.

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

=== Test Set Evaluation (Initial Sklearn Model) ===
Accuracy: 0.6996
              precision    recall  f1-score   support

  BACKGROUND       0.61      0.37      0.46      3621
 CONCLUSIONS       0.61      0.55      0.57      4571
     METHODS       0.68      0.88      0.77      9897
   OBJECTIVE       0.72      0.09      0.16      2333
     RESULTS       0.77      0.85      0.81      9713

    accuracy                           0.70     30135
   macro avg       0.68      0.55      0.56     30135
weighted avg       0.69      0.70      0.67     30135

Macro-averaged F1 score: 0.5555

Starting Hyperparameter Tuning on Development Set...
Fitting 3 folds for each of 18 candidates, totalling 54 fits
Grid search complete.
Best Parameters: {'nb__alpha': 0.5, 'tfidf__min_df': 5, 'tfidf__ngram_range': (1, 2)}
Best F1 Score: 0.6069
```

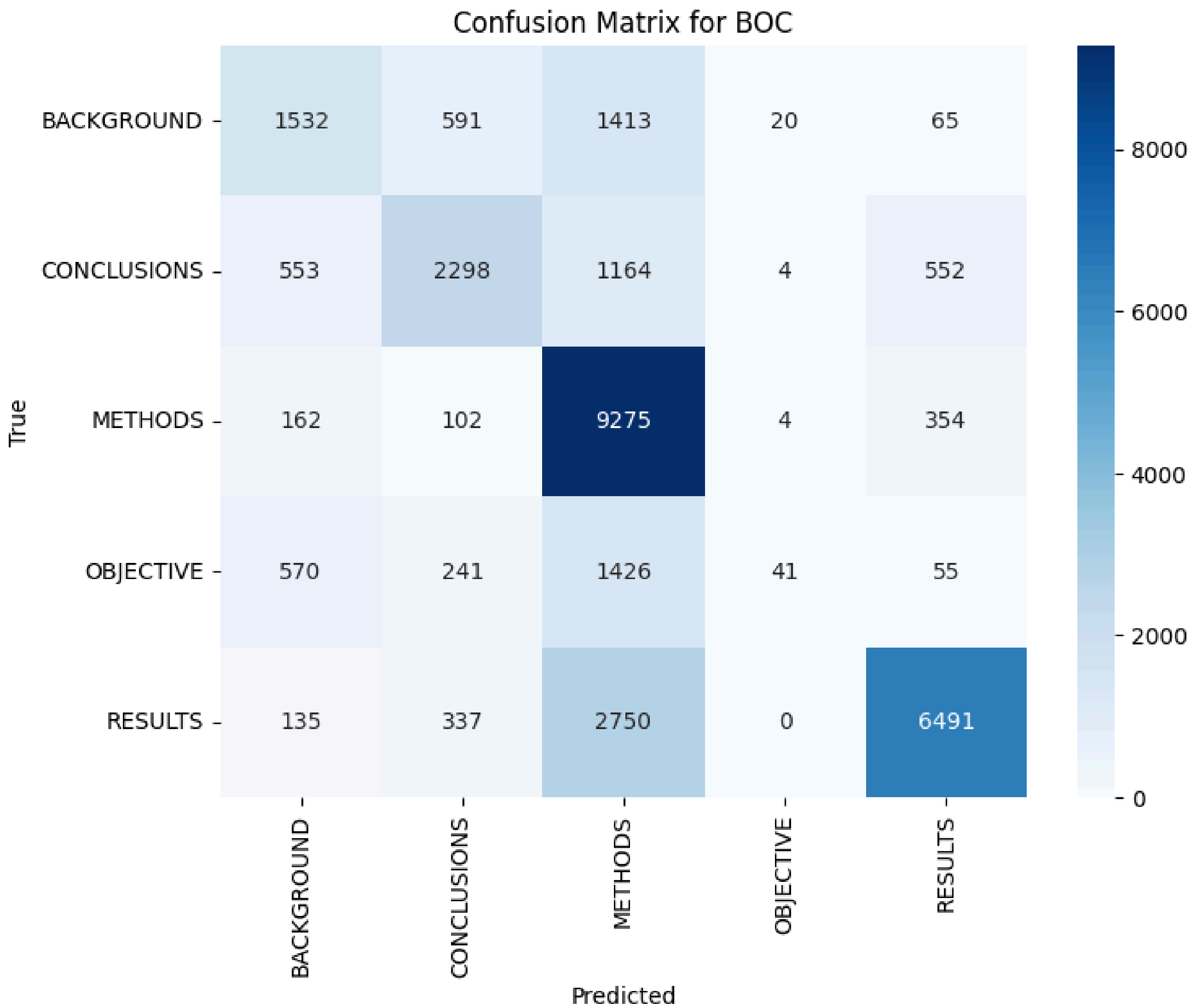


Part C: Bayes Optimal Classifier Approximation

The BOC was approximated using a soft-voting ensemble trained on a subset of the data.

```
My SRN isPES2UG23CS368
Using dynamic sample size: 10368
Actual sampled training set size used: 10368
Using 10368 samples for training base models.
```

```
=== Final Evaluation: Bayes Optimal Classifier (Hard Voting) ===
BOC Accuracy: 0.6516
BOC Macro F1 Score: 0.5068
              precision    recall  f1-score   support

  BACKGROUND       0.52      0.42      0.47      3621
 CONCLUSIONS       0.64      0.50      0.56      4571
     METHODS       0.58      0.94      0.72      9897
   OBJECTIVE       0.59      0.02      0.03      2333
     RESULTS       0.86      0.67      0.75      9713

    accuracy                           0.65     30135
   macro avg       0.64      0.51      0.51     30135
weighted avg       0.67      0.65      0.62     30135
```

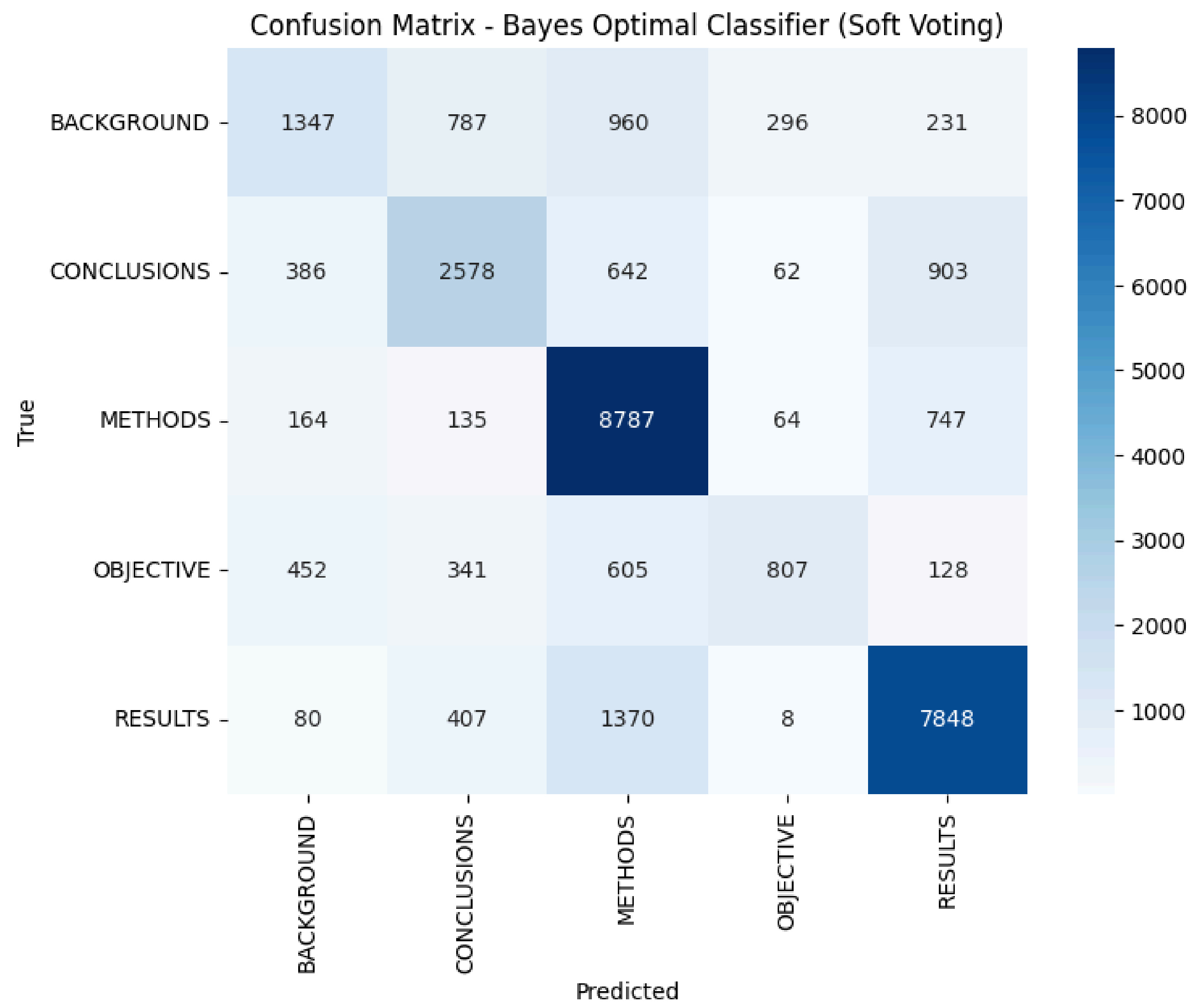


```
Using dynamic sample size: 10368
Actual sampled training set size used: 10368

Training all base models...
Training NaiveBayes
Training LogisticRegression
Training RandomForest
/Users/polarhive/.local/repos/pesu/ML_F_PES2UG23CS368_Nathan_Paul/.venv/lib/python3.14/site-packages/sklearn/linear_model/_logistic.py:127
  warnings.warn(
/Users/polarhive/.local/repos/pesu/ML_F_PES2UG23CS368_Nathan_Paul/.venv/lib/python3.14/site-packages/sklearn/linear_model/_logistic.py:129
  warnings.warn(
Training DecisionTree
Training KNN
All base models trained.

Using 8294 samples for sub-training and 2074 for validation (likelihood calc).
Evaluating log-likelihood on validation set for NaiveBayes...
  Log-likelihood: -1649.1614
Evaluating log-likelihood on validation set for LogisticRegression...
  Log-likelihood: -1474.5076
Evaluating log-likelihood on validation set for RandomForest...
  Log-likelihood: -1737.4064
Evaluating log-likelihood on validation set for DecisionTree...
  Log-likelihood: -2500.6371
Evaluating log-likelihood on validation set for KNN...
  Log-likelihood: -2694.7877

Posterior weights P(h | D) (normalized):
  NaiveBayes: 0.0000
  LogisticRegression: 1.0000
  RandomForest: 0.0000
  DecisionTree: 0.0000
  KNN: 0.0000
```

```
Predicting on test set...

=== Final Evaluation: Bayes Optimal Classifier (Soft Voting) ===
```


```
=== Final Evaluation: Bayes Optimal Classifier (Soft Voting) ===
```


```
Accuracy: 0.7090
Macro F1 : 0.6147

Classification Report:
              precision    recall  f1-score   support

  BACKGROUND       0.55      0.37      0.45      3621
 CONCLUSIONS       0.61      0.56      0.58      4571
     METHODS       0.71      0.89      0.79      9897
   OBJECTIVE       0.65      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
```



---

4. Discussion

A comparison of the three models reveals interesting insights into the dataset and feature representation:

1. **Part A (Scratch MNB):** This model achieved the highest performance, with an **Accuracy of 0.7537** and a **Macro F1 of 0.6836**. This model used `CountVectorizer` (simple word counts) with an `ngram_range` of (1, 2) on the full training set.
2. **Part B (Tuned Sklearn MNB):** The hyperparameter-tuned Sklearn model, which used `TfidfVectorizer`, only achieved a **best F1 score of 0.6069** on the development set. The initial, untuned Sklearn model performed even worse on the test set (Acc: 0.6996, F1: 0.5555). This suggests that for this specific classification task, raw word counts (`CountVectorizer`) are a more effective feature representation than TF-IDF (term frequency-inverse document frequency).
3. **Part C (BOC Approximation):** The BOC (Soft Voting) model achieved an **Accuracy of 0.7090** and a **Macro F1 of 0.6147**. This was a slight improvement over the single best Sklearn model from Part B.

- An important observation is that the posterior weight calculation assigned a weight of **1.0000 to Logistic Regression** and 0.0000 to all other models. This means the soft vote was not a true ensemble but effectively just the output of the Logistic Regression model.
- This BOC model performed significantly worse than the scratch model from Part A. This is primarily because the Part C ensemble was trained on a **small sample of the data (10,368)**, whereas the Part A model was trained on the **full dataset (180,040)**. The reduced data size was a major performance bottleneck.

In conclusion, the custom Naive Bayes classifier built from scratch on the full dataset with count-based features was the most effective model for this task.