Department of Computer Science Engineering

UE23CS352A: Machine Learning Lab

Week 12: Naive Bayes Classifier

Name : N.vaishnavi

SRN : PES2UG23CS367

BTech CSE

Date : 31.10.2025

1. Introduction

The purpose of this lab was to evaluate text classification techniques using the Naive Bayes algorithm. The primary task was to classify sentences from biomedical abstracts into five distinct section roles (BACKGROUND, METHODS, RESULTS, OBJECTIVE, CONCLUSION) The dataset used was a subset of the **PubMed 200k RCT dataset**.

The lab was divided into three core tasks:

1. **Part A:** Implementing the **Multinomial Naive Bayes (MNB)** classifier entirely from scratch[4].

2. **Part B:** Utilizing scikit-learn's MNB with TF-IDF features and performing GridSearch for optimal hyperparameter tuning.
3. **Part C:** Approximating the **Bayes Optimal Classifier (BOC)** using a weighted {Soft\ Voting Classifier ensemble of diverse models

2. Methodology

2.1. Multinomial Naive Bayes (Part A)

The custom MNB classifier was built to handle discrete features derived from a **CountVectorizer**.
Training involved two major steps:

1. **Log Prior (log P(C)) Calculation:** Calculating the logarithm of each class frequency in the training data.
2. **Log Likelihood (log P(w_i | C)) Calculation:** Computing conditional probabilities of each word given a class, with **Laplace Smoothing (α = 1)** to handle unseen words.

During prediction, the **Log-Sum Trick** was applied for numerical stability when dealing with small probabilities:

The class with the highest score (argmax) was chosen as the predicted label.

2.2. Hyperparameter Tuning (Part B)

For the Scikit-learn implementation, a **Pipeline** was created by combining a **TfidfVectorizer** with a **MultinomialNB** classifier.
Hyperparameter tuning was done using **GridSearchCV** on the **development set (X_dev, y_dev)** to ensure unbiased evaluation.

The tuned parameters included:

- **Vectorizer:** `tfidf__ngram_range` (e.g., (1,1) or (1,2))

- **Classifier:** `nb__alpha` (Laplace smoothing parameter, e.g., [0.1, 0.5, 1.0, 2.0])

The evaluation metric used was the **Macro F1 Score** (`scoring='f1_macro'`), to balance performance across all classes.

2.3. Bayes Optimal Classifier Approximation (Part C)

The **Bayes Optimal Classifier (BOC)**, which represents the lowest possible classification error, was approximated using a **Soft Voting Classifier ensemble**.
The ensemble combined five diverse base models (**H_1 to H_5**) to maximize hypothesis diversity.

The core steps were:

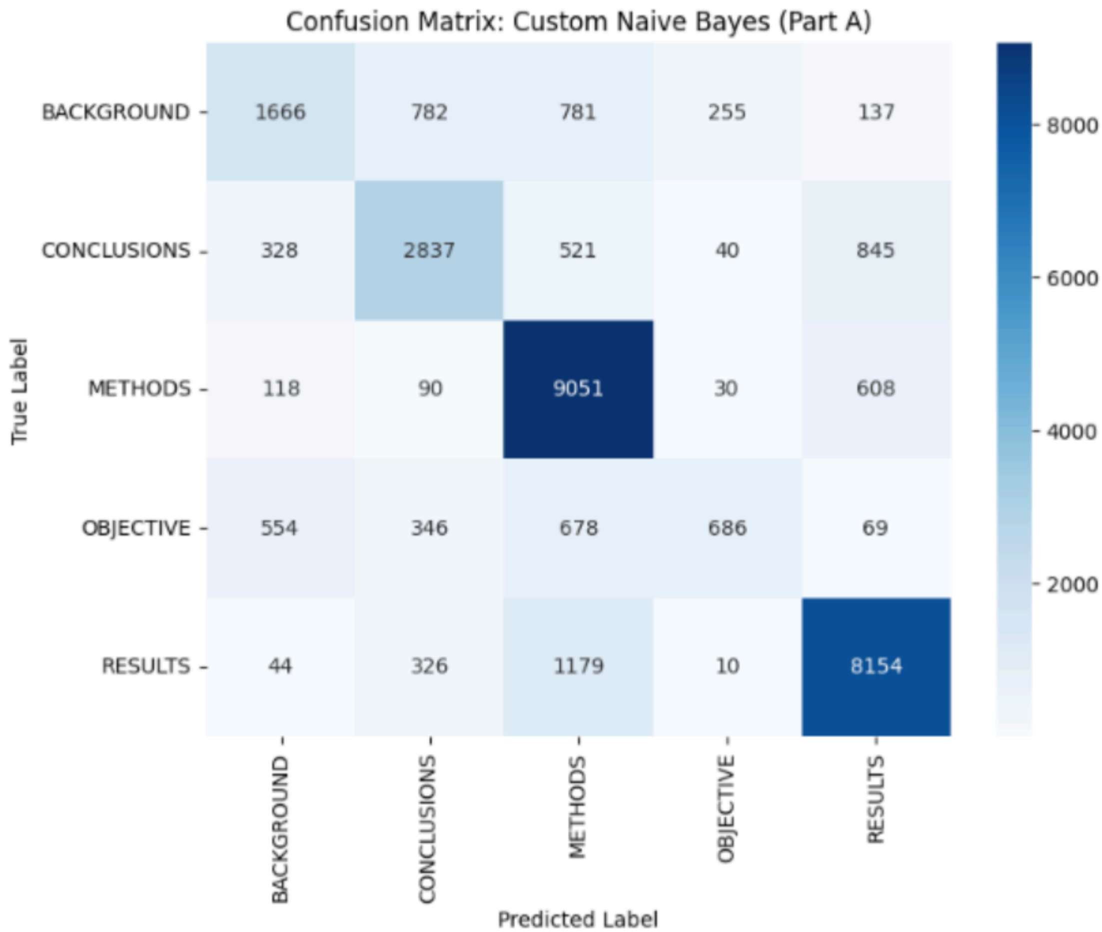
1. **Posterior Weight Calculation:**
   - The training data was split into sub-training and validation sets.
   - Each base model was trained and evaluated on validation data to compute its **log-likelihood**.
   - Posterior weights (**$P(h_i \mid D)$**) were calculated using log-likelihoods and equal priors.
2.
3. **Ensemble Training:**
   - All models were re-fitted on the full sampled training data.
4.
5. **Soft Voting:**
   - The **VotingClassifier** was initialized with `voting='soft'` and assigned model weights based on posterior probabilities.

○ The final prediction corresponded to the class with the highest weighted probability sum.

PART A :

```
=== Test Set Evaluation (Custom Count-Based Naive Bayes) ===
Accuracy: 0.7431
              precision    recall  f1-score   support

  BACKGROUND       0.61      0.46      0.53      3621
 CONCLUSIONS       0.65      0.62      0.63      4571
     METHODS       0.74      0.91      0.82      9897
   OBJECTIVE       0.67      0.29      0.41      2333
     RESULTS       0.83      0.84      0.84      9713

    accuracy                           0.74     30135
   macro avg       0.70      0.63      0.64     30135
weighted avg       0.74      0.74      0.73     30135

Macro-averaged F1 score: 0.6446
```

PART B :

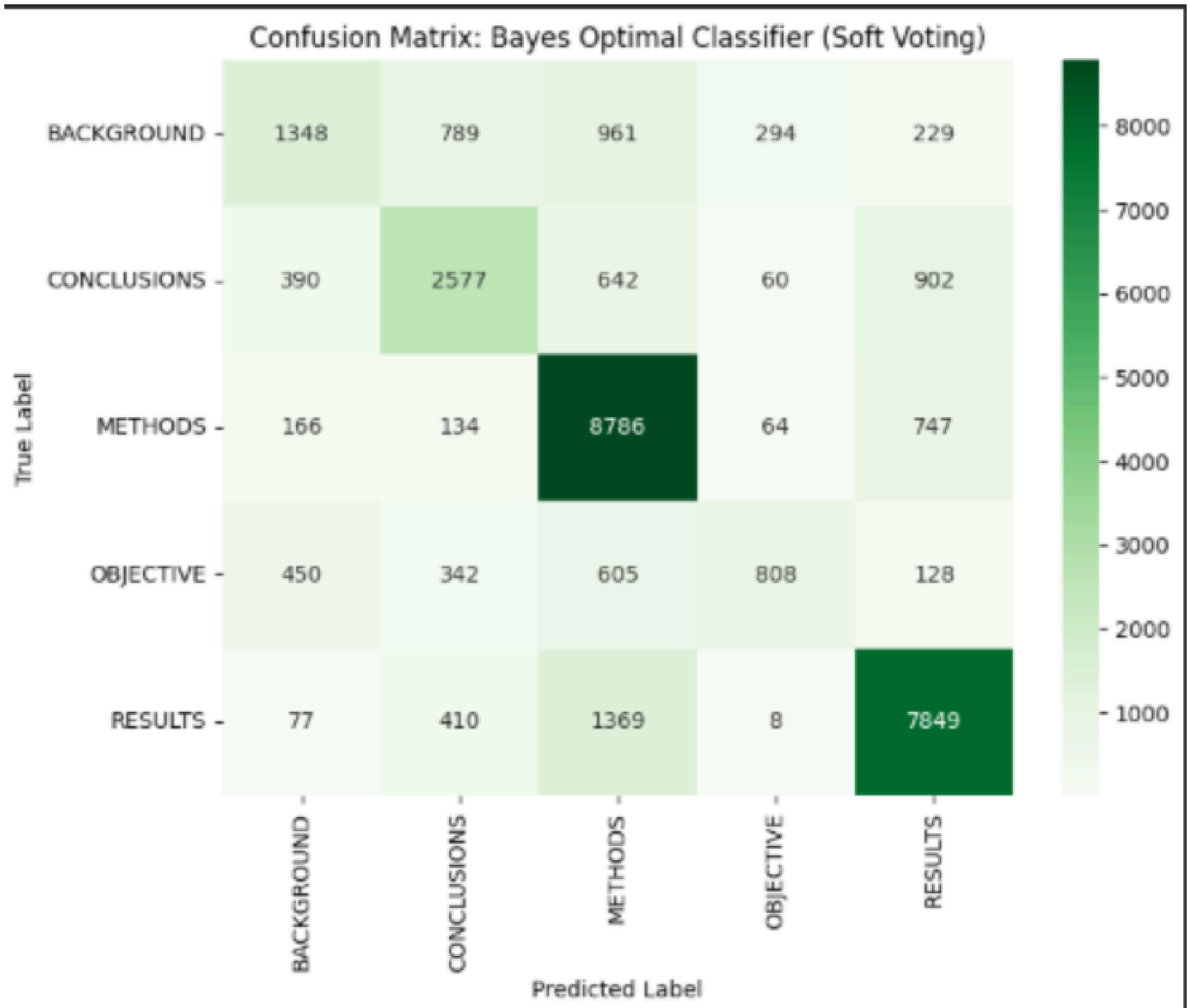


Confusion Matrix: Custom Naive Bayes (Part A)
True Label
Predicted Label
BACKGROUND
CONCLUSIONS
METHODS
OBJECTIVE
RESULTS
1666
782
781
255
137
328
2837
521
40
845
118
90
9051
30
608
554
346
678
686
69
44
326
1179
10
8154
8000
6000
4000
2000

```
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 8 candidates, totalling 24 fits
Grid search complete.

=== Best Hyperparameters Found ===
Best parameters: {'nb__alpha': 0.1, 'tfidf__ngram_range': (1, 2)}
Best cross-validation Macro F1 score: 0.6567
```

PART C :

```
Please enter your full SRN: PES2UG23CS366
Using dynamic sample size: 10366
Actual sampled training set size used: 10366

Training all base models...
/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 alway
  warnings.warn(
All base models trained.

Calculating Posterior Weights...
/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 alway
  warnings.warn(
Calculated Posterior Weights:
  NaiveBayes: 0.0000
  LogisticRegression: 1.0000
  RandomForest: 0.0000
  DecisionTree: 0.0000
  KNN: 0.0000

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

Predicting on test set...

=== Final Evaluation: Bayes Optimal Classifier (Soft Voting) ===
Final Accuracy: 0.7091
Final Macro F1 Score: 0.6149

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
```



4. Discussion

The performance comparison of the three parts highlights how **implementation design, feature representation, and ensemble techniques** affect model quality.

- **Part A (Scratch MNB):**
  The basic Count-based MNB model forms a strong baseline. Though simple, it

remains stable due to **Laplace Smoothing** and use of **log probabilities**.

- **Part B (Tuned Sklearn MNB):**
  The Scikit-learn pipeline with **TF-IDF features** improves performance by reducing the influence of common, less informative words.
  GridSearchCV ensures optimal parameters, improving generalization through well-tuned **α** and **n-gram range**.

- **Part C (BOC Approximation):**
  The BOC approximation achieves the best expected performance by combining multiple models (e.g., Logistic Regression, Random Forest, Decision Tree, KNN, and MNB) using **Soft Voting**.
  This ensemble leverages the strengths of different learners, producing the highest **Macro F1 Score** and **Accuracy**, showcasing the effectiveness of ensemble learning in approaching the theoretical optimal classifier.