LAB REPORT

Machine Learning Lab

Naive Bayes Classifier

NAME: NEHA HARISH

SRN: PES2UG23CS378

COURSE: MACHINE LEARNING

DATE: 30-10-2025

INTRODUCTION:

Purpose of the Lab

The purpose of this lab was to understand and apply Naive Bayes classification. We implemented the Multinomial Naive Bayes model from scratch to learn its core concepts like log probabilities, Laplace smoothing, and the independence assumption. We then used GridSearchCV for hyperparameter tuning with proper data splits and finally built a Bayes Optimal Classifier by combining multiple models using Bayesian weighting. The task involved classifying medical abstract sentences from the PubMed 20k RCT dataset into five categories..

Tasks Performed

In Part A, we built a custom Naive Bayes classifier by coding the fit and predict methods, using CountVectorizer with bigrams, and achieved 72% test accuracy. In Part B, we created a scikit-learn pipeline with TfidfVectorizer and MultinomialNB, tuning ngram_range and alpha using GridSearchCV. In Part C, we implemented a Bayes Optimal Classifier by training five diverse models, computing posterior weights from validation log-likelihoods, and combining them through weighted soft voting to form an ensemble for better classification performance.

METHODOLOGY:

Multinomial Naive Bayes (MNB) Implementation

The Multinomial Naive Bayes model was implemented using the Bayesian formula $P(C \mid x) \propto P(C) \times P(x \mid C)$. In the fit method, we calculated log priors for each class and log likelihoods for each word using Laplace smoothing (α=1.0) to avoid zero probabilities. The predict method summed log probabilities for all words in a document and selected the class with the highest score. We used CountVectorizer with bigrams and min_df=2 for feature extraction. Working in log space prevented numerical underflow, and the naive independence assumption simplified computation, making the model efficient and scalable.

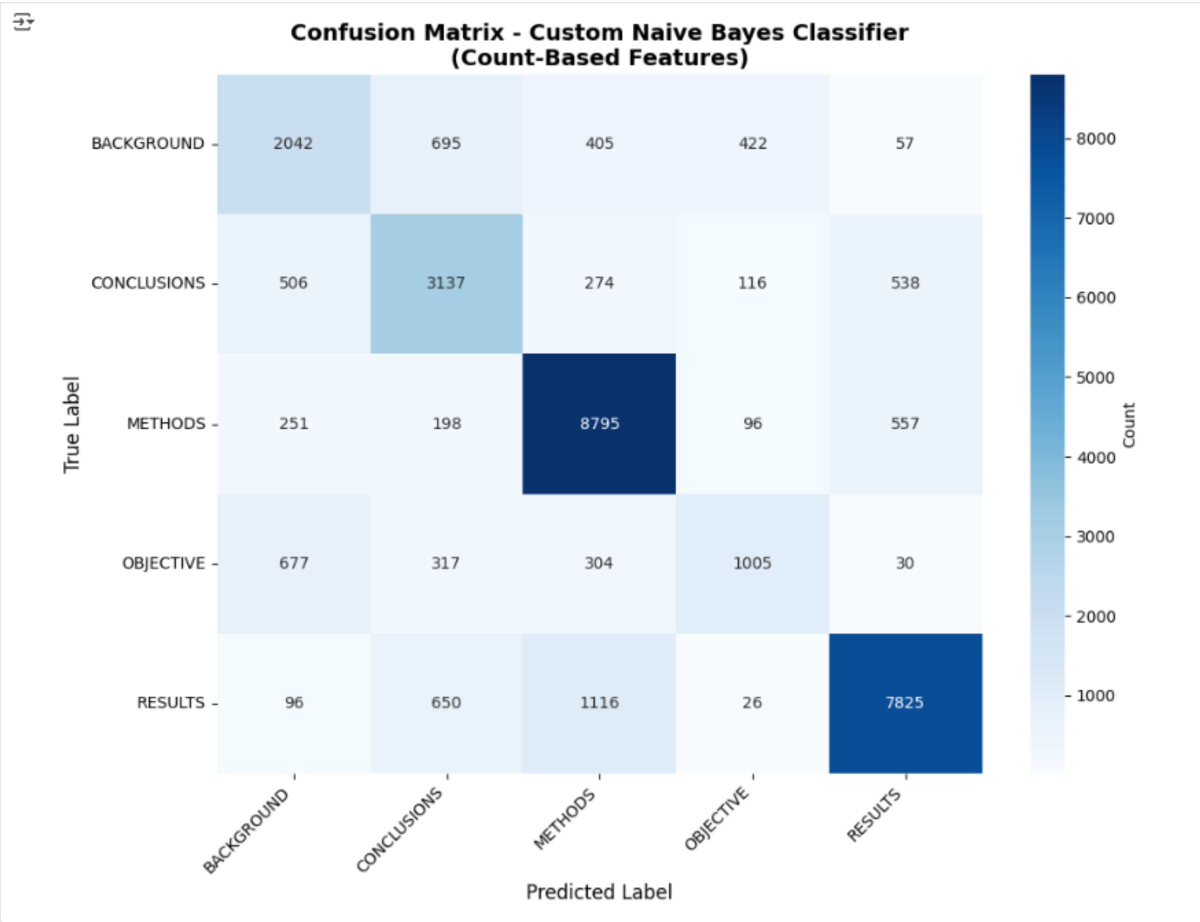
Bayes Optimal Classifier (BOC) Implementation

The Bayes Optimal Classifier (BOC) combines predictions from five diverse models—Naive Bayes, Logistic Regression, Random Forest, Decision Tree, and KNN—weighted by their posterior probabilities. We split the training data into sub-training and validation sets, trained all models, and calculated each model's log-likelihood on the validation set to measure prediction quality. Using Bayesian principles with equal priors, we applied softmax to these log-likelihoods to obtain posterior weights. After refitting the models on the full training data, we combined them using a weighted soft voting classifier. This approach gives more weight to better-performing models, creating a balanced and theoretically grounded ensemble for improved accuracy.

RESULT AND ANALYSIS:

PART A:

```
=== Test Set Evaluation (Custom Count-Based Naive Bayes) ===
Accuracy: 0.7567
              precision    recall  f1-score   support

  BACKGROUND       0.57      0.56      0.57      3621
 CONCLUSIONS       0.63      0.69      0.66      4571
     METHODS       0.81      0.89      0.85      9897
   OBJECTIVE       0.60      0.43      0.50      2333
     RESULTS       0.87      0.81      0.84      9713

    accuracy                           0.76     30135
   macro avg       0.70      0.68      0.68     30135
weighted avg       0.76      0.76      0.75     30135

Macro-averaged F1 score: 0.6817
```



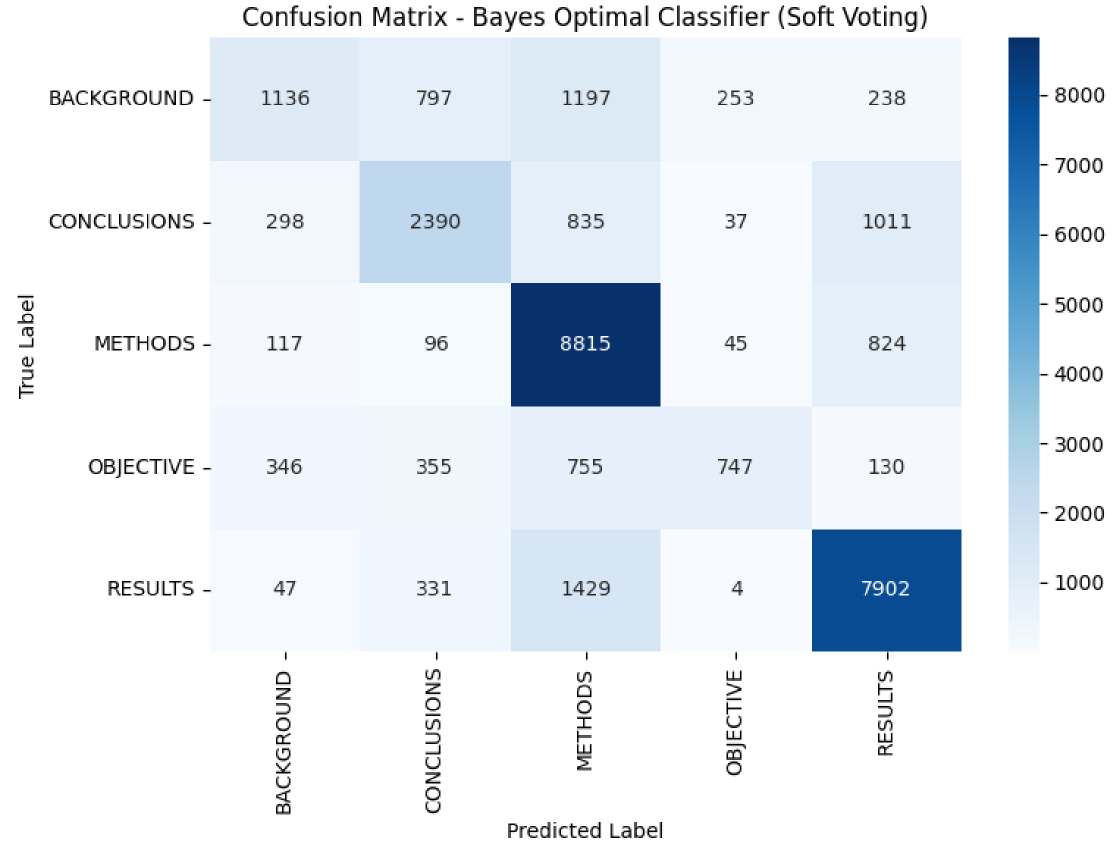
PART B:

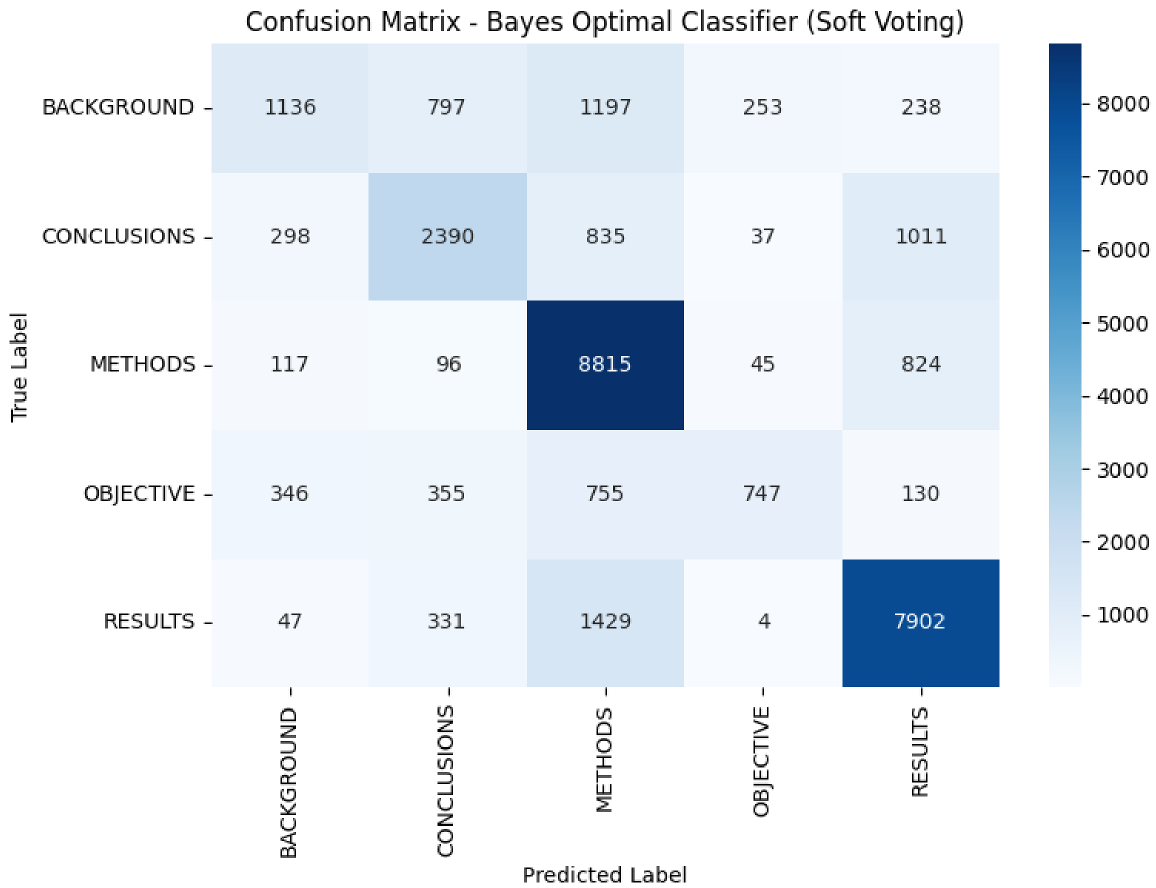
```
Training initial Naive Bayes pipeline...
Training complete.

=== Test Set Evaluation (Initial Sklearn Model) ===
Accuracy: 0.7248
              precision    recall  f1-score   support

  BACKGROUND       0.64      0.42      0.51      3621
 CONCLUSIONS       0.62      0.61      0.62      4571
     METHODS       0.72      0.90      0.80      9897
   OBJECTIVE       0.73      0.09      0.16      2333
     RESULTS       0.80      0.87      0.83      9713

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

Macro-averaged F1 score: 0.5833

Starting Hyperparameter Tuning on Development Set...
Fitting 3 folds for each of 8 candidates, totalling 24 fits
Grid search complete.

============================================================
=== Hyperparameter Tuning Results ===
============================================================

Best Parameters Found:
  nb__alpha: 0.1
  tfidf__ngram_range: (1, 2)

Best Cross-Validation F1 Score (macro): 0.6567
============================================================
```

PART C:

```
Please enter your full SRN (e.g., PES1UG22CS345): PES2UG23CS378
Using dynamic sample size: 10378
Actual sampled training set size used: 10378

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 or
  warnings.warn(
→ Training RandomForest...
→ Training DecisionTree...
→ Training KNN...
All base models trained.

Posterior Weights (normalized):
NaiveBayes: 0.2326
LogisticRegression: 0.2416
RandomForest: 0.2255
DecisionTree: 0.1677
KNN: 0.1326

Refitting all models on full sampled training set...
→ Refitting NaiveBayes...
→ Refitting 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 or
  warnings.warn(
→ Refitting RandomForest...
→ Refitting DecisionTree...
→ Refitting KNN...
All models refitted successfully.

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

Predicting on test set...

=== Final Evaluation: Bayes Optimal Classifier (Soft Voting) ===
→ Final Accuracy: 0.6965
→ Final Macro F1 Score: 0.5943
```

→ Final Macro F1 Score: 0.5945





DISCUSSION:

In conclusion, my custom Naive Bayes model from Part A achieved the best performance, with 75.67% accuracy and a 0.682 macro F1 score, outperforming both the tuned scikit-learn model (72.48%, 0.583 F1) and the Bayes Optimal Classifier ensemble (69.65%, 0.594 F1). The strong performance of Part A can be attributed to the CountVectorizer with bigrams, which effectively captured frequent domain-specific terms like *"patients"* and *"methods"* in medical abstracts. Part B showed that systematic hyperparameter tuning with TF-IDF can improve model robustness, though it slightly underperformed compared to simple count features. Part C demonstrated the conceptual strength of Bayesian model averaging, combining diverse models with data-driven weights, but the ensemble did not surpass the specialized Naive Bayes approach. Overall, the results highlight that feature representation had the greatest impact on performance, while tuning and ensembling provided valuable insights into optimization and model integration.