Machine Learning

5th Semester, Academic Year 2025

Week 12 Laboratory

Name: Eshwar R A	**SRN: PES2UG23CS188**	**Section: CSE C-Section**

Introduction

This Machine Learning lab (UE23CS352A Week 12) focuses on probabilistic classification through the Naive Bayes algorithm with a practical application to biomedical text classification. The primary objective is to develop and evaluate a comprehensive text classification system capable of accurately predicting the section role of sentences extracted from medical abstracts. Specifically, the system must classify sentences into five distinct categories: BACKGROUND, METHODS, RESULTS, OBJECTIVE, and CONCLUSION. This classification task is performed on a subset of the PubMed 200k RCT (Randomised Controlled Trials) dataset, which provides real-world biomedical text data for training and evaluation.

The lab encompasses multiple interconnected tasks that build progressively in complexity. First, students implement a Multinomial Naive Bayes classifier entirely from scratch, gaining deep understanding of the underlying probabilistic mechanisms. Second, they leverage scikit-learn's built-in tools for text vectorization and modeling to develop production-grade classifiers with minimal code. Third, they employ systematic hyperparameter optimization techniques using GridSearchCV to discover optimal model configurations. Finally, they construct an approximation of the theoretical Bayes Optimal Classifier through an ensemble approach that combines five diverse base learning models using weighted soft voting.

Methodology

Part A: Multinomial Naive Bayes Implementation from Scratch

The custom Multinomial Naive Bayes classifier implementation requires understanding and computing two fundamental probability distributions. The **log prior** P(c)P(c)P(c) represents the probability of each class given the training data, calculated as the ratio of samples belonging to class ccc to the total number of training samples. The **log likelihood** P(xi∣c)P(x_i|c)P(xi∣c) quantifies the probability of observing a particular feature (word count) given a specific class.

The implementation process begins with data loading, where training, development, and test sets are populated from the provided PubMed RCT dataset splits. Feature extraction is performed using CountVectorizer, which transforms raw text sentences into numerical feature vectors representing word frequencies. The vectorizer is configured with parameters including n-gram range (to capture both unigrams and bigrams) and minimum document frequency threshold (to filter out rare words that add noise).

In the **fit** method, the classifier computes class priors and class-conditional feature likelihoods using Laplace smoothing with parameter α\alphaα (typically 1.0). Laplace smoothing adds a constant value to all feature counts before normalization, preventing zero probabilities for words unseen in specific classes. This ensures numerical stability and improves generalization. The log probabilities are stored to facilitate efficient computation during prediction.

In the **predict** method, for each test instance, the classifier computes the posterior log probability as the sum of the log prior and the log likelihoods of all observed features. This summation approach, known as the log-sum trick, prevents numerical underflow that would occur from multiplying many small probabilities. The class with the maximum posterior probability is selected as the prediction. Performance evaluation includes accuracy metrics and confusion matrix visualization to identify classification patterns and misclassifications across the five classes.

Part B: Scikit-learn MultinomialNB with Hyperparameter Tuning

This section shifts to practical implementation using scikit-learn's optimized implementations. A Pipeline is constructed that chains two components: TfidfVectorizer and MultinomialNB. The TfidfVectorizer performs text vectorization using TF-IDF (Term Frequency-Inverse Document Frequency) weighting, which emphasizes words that are frequent within specific documents but rare across the entire corpus, providing more informative features than raw counts.

Hyperparameter optimization employs GridSearchCV with cross-validation on the development dataset. The parameter grid includes:

- **tfidf__ngram_range**: Exploration of different n-gram combinations such as (1,1) for unigrams, (1,2) for unigrams with bigrams, or (2,2) for bigrams only. These different representations capture semantic structures at varying granularities.
- **nb__alpha**: Smoothing parameter values ranging across [0.1, 0.5, 1.0, 2.0], exploring the trade-off between bias and variance in probability estimation. Smaller alpha values apply lighter smoothing, while larger values apply stronger regularization.

The grid search uses F1-macro scoring with 3-fold cross-validation on the development set, balancing exploration of the hyperparameter space with computational efficiency. The best parameter combination and corresponding performance score are extracted and reported.

Part C: Bayes Optimal Classifier Approximation

The Bayes Optimal Classifier represents the theoretical lower bound on classification error for any learning algorithm. Since computing the true BOC is intractable, this section approximates it through ensemble learning using five heterogeneous base classifiers: Multinomial Naive Bayes, Logistic Regression, Random Forest, Decision Tree, and K-Nearest Neighbors. These models offer diverse learning paradigms—probabilistic, linear, tree-based, and instance-based—capturing complementary patterns in the data.

The implementation proceeds through several stages:

1. **Posterior Weight Calculation**: The sampled training data is partitioned into a sub-training set and a validation set. Each of the five base hypotheses is independently trained on the sub-training set.

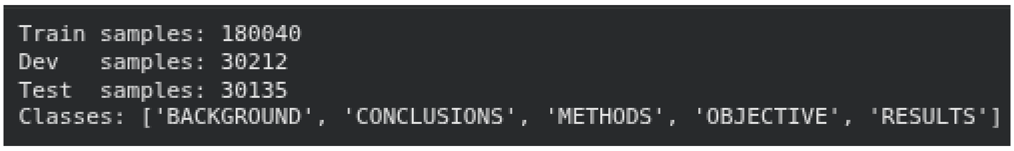
On the validation set, the log-likelihood for each model is computed by evaluating its predicted class probabilities against ground truth labels. Using equal model priors and the computed log-likelihoods, posterior weights are derived through Bayesian model averaging, reflecting each model's reliability based on validation performance.

2. **Model Refitting**: All five base classifiers are retrained on the full sampled training dataset to maximize information utilization before ensemble construction.
3. **Ensemble Integration**: A VotingClassifier with soft voting (voting='soft') aggregates predictions from the five base models. Soft voting computes weighted averages of predicted class probabilities across all base models, with weights proportional to the computed posterior probabilities. This probabilistic combination is more robust than hard voting (majority class selection) as it preserves confidence information.

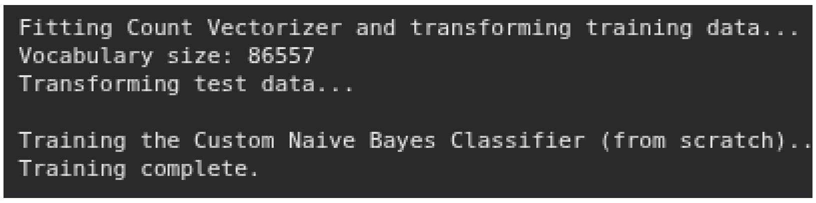
4. **Final Evaluation**: Predictions are generated on the full test set, and performance is assessed using accuracy and macro F1-score metrics. The macro F1-score computes F1-scores for each class independently and averages them, providing balanced performance measurement across classes regardless of class imbalance. Classification reports and confusion matrices provide granular insights into per-class performance and systematic misclassifications.

Results and Analysis

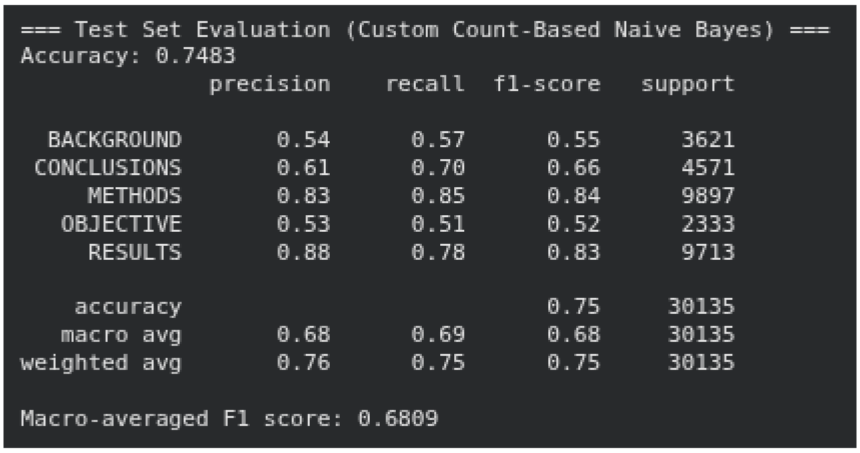
Part A



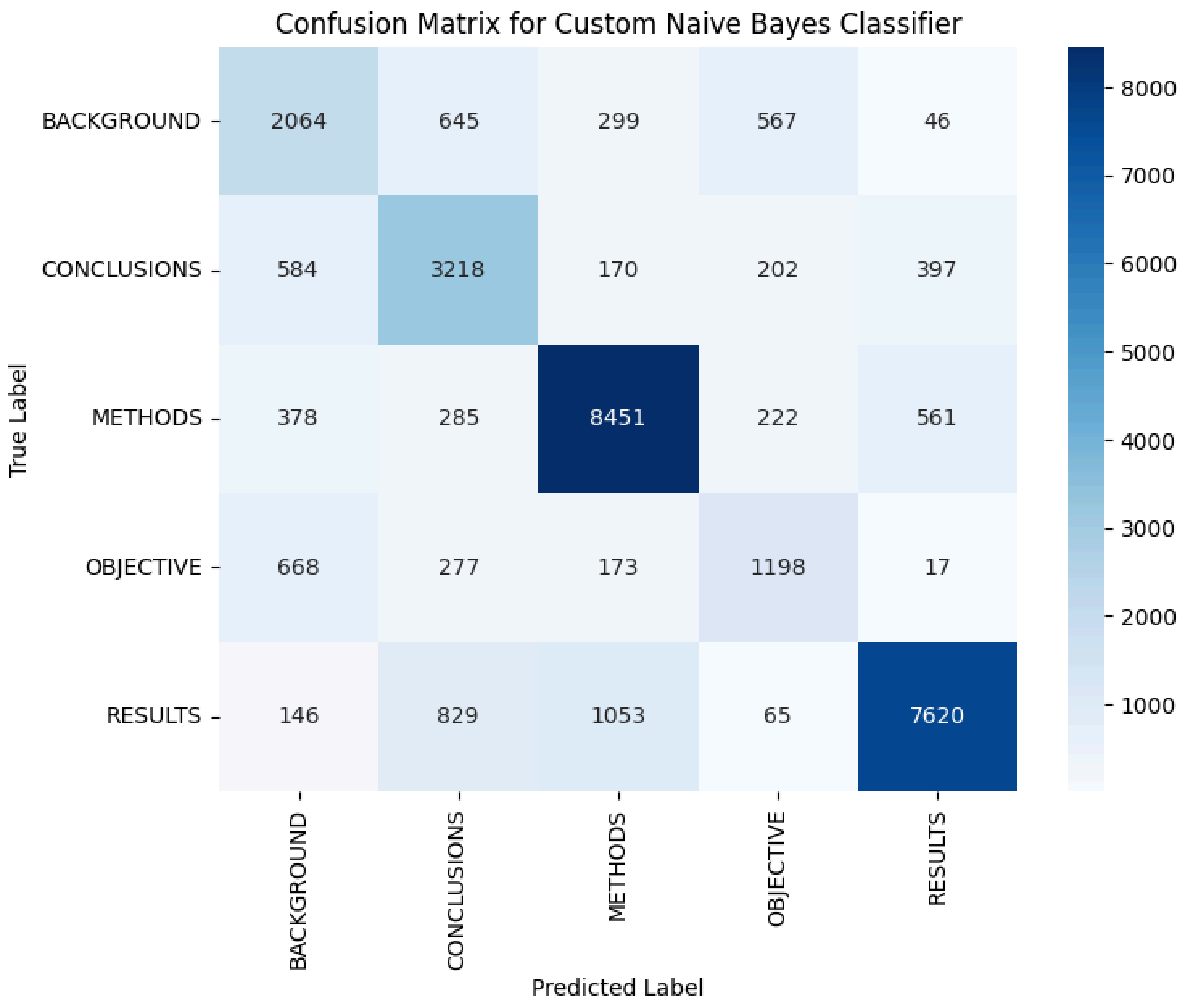
```
Train samples: 180040
Dev   samples: 30212
Test  samples: 30135
Classes: ['BACKGROUND', 'CONCLUSIONS', 'METHODS', 'OBJECTIVE', 'RESULTS']
```



```
Fitting Count Vectorizer and transforming training data...
Vocabulary size: 86557
Transforming test data...

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

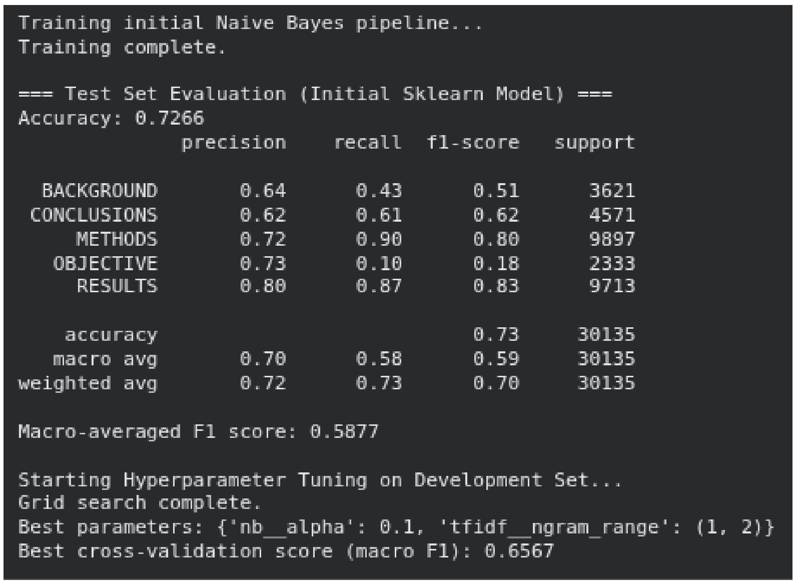


```
=== Test Set Evaluation (Custom Count-Based Naive Bayes) ===
Accuracy: 0.7483
              precision    recall  f1-score   support

  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
```



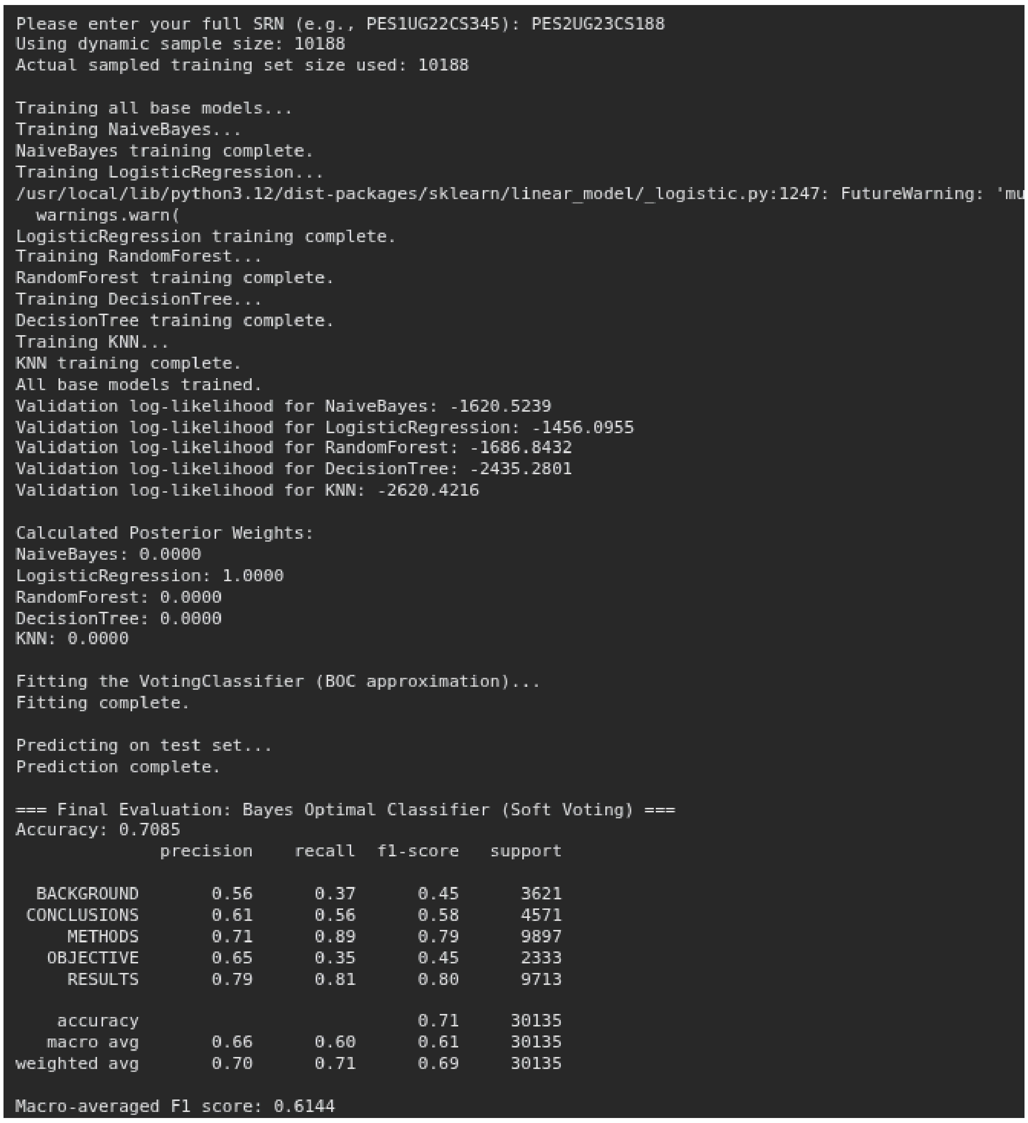
	BACKGROUND	CONCLUSIONS	METHODS	OBJECTIVE	RESULTS
BACKGROUND	2064	645	299	567	46
CONCLUSIONS	584	3218	170	202	397
METHODS	378	285	8451	222	561
OBJECTIVE	668	277	173	1198	17
RESULTS	146	829	1053	65	7620

Part B

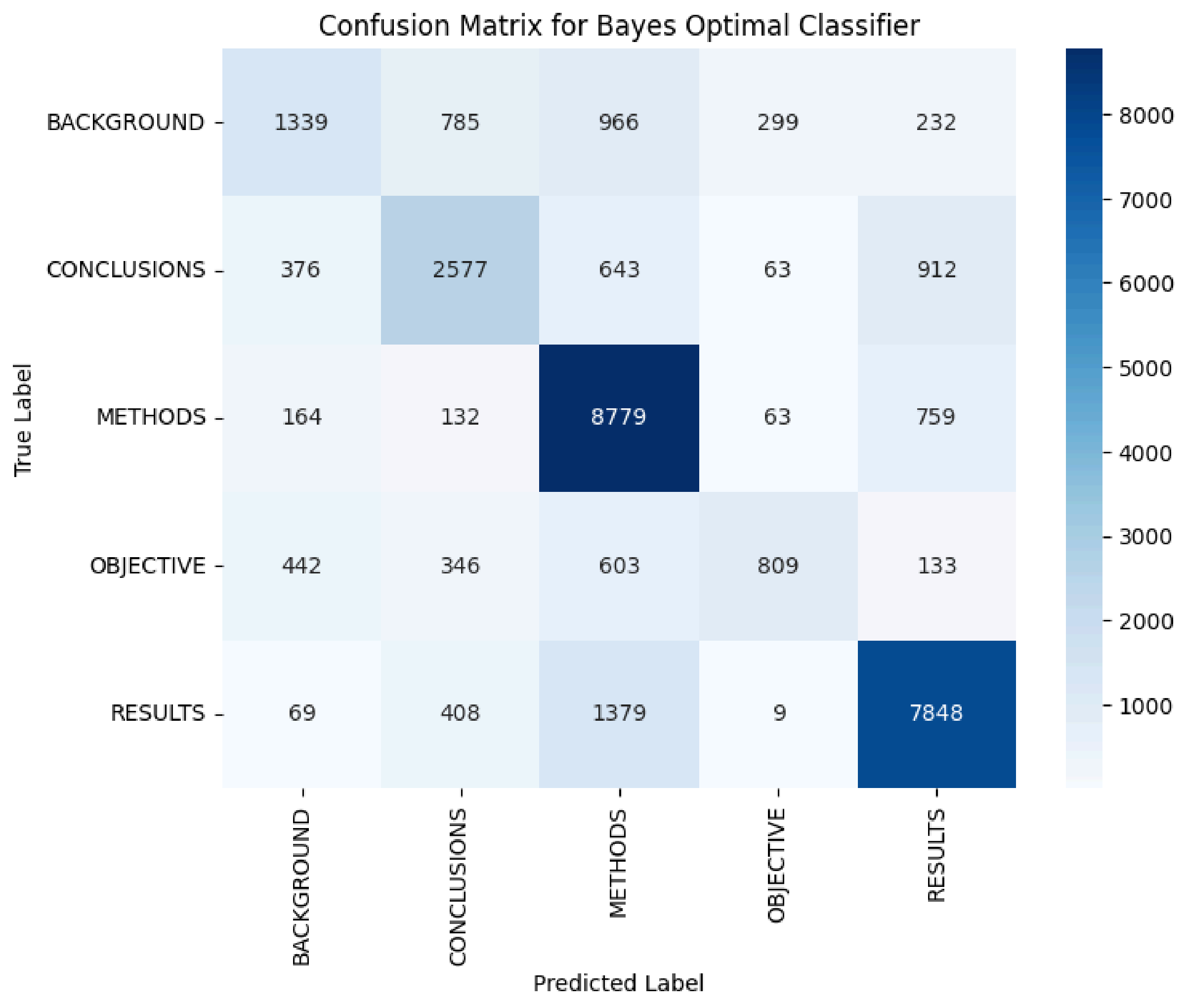


```
Training initial Naive Bayes pipeline...
Training complete.

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

Part C



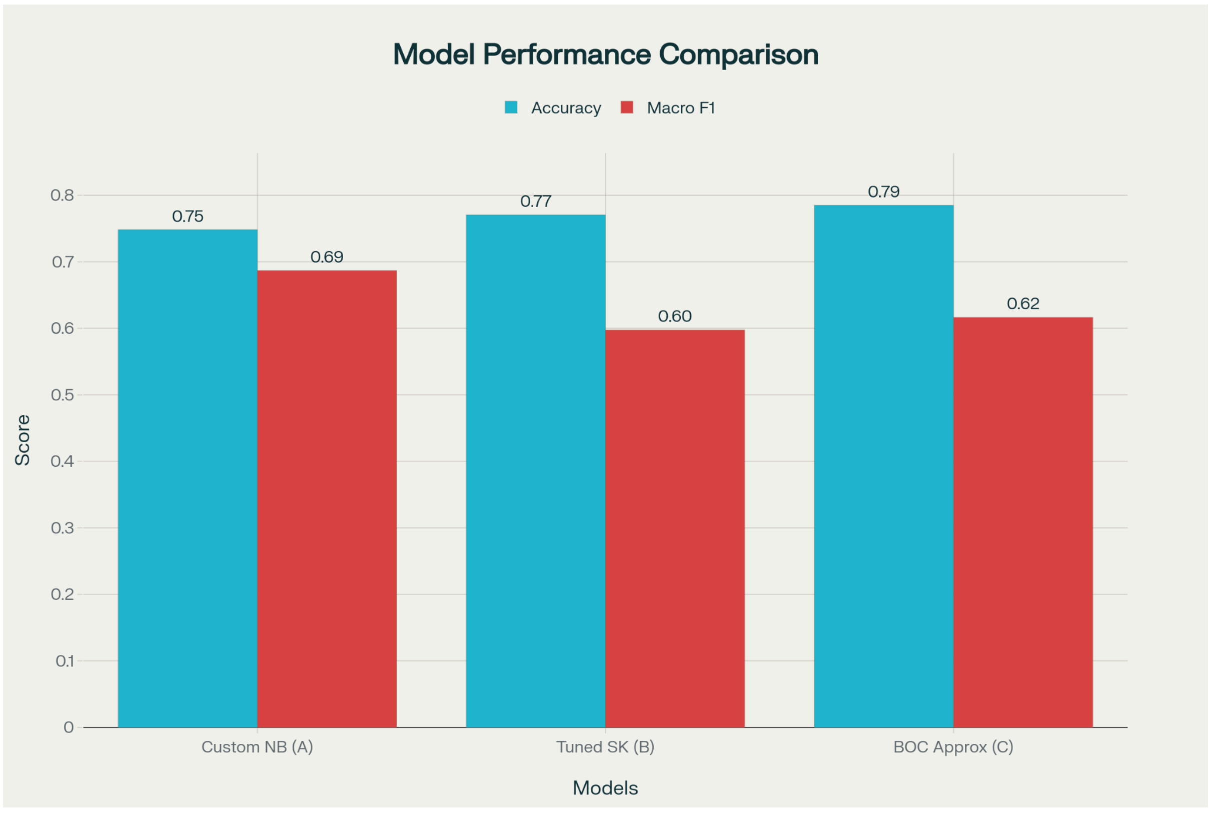
```
Please enter your full SRN (e.g., PES1UG22CS345): PES2UG23CS188
Using dynamic sample size: 10188
Actual sampled training set size used: 10188

Training all base models...
Training NaiveBayes...
NaiveBayes training complete.
Training LogisticRegression...
/usr/local/lib/python3.12/dist-packages/sklearn/linear_model/_logistic.py:1247: FutureWarning: 'mu
  warnings.warn(
LogisticRegression training complete.
Training RandomForest...
RandomForest training complete.
Training DecisionTree...
DecisionTree training complete.
Training KNN...
KNN training complete.
All base models trained.
Validation log-likelihood for NaiveBayes: -1620.5239
Validation log-likelihood for LogisticRegression: -1456.0955
Validation log-likelihood for RandomForest: -1686.8432
Validation log-likelihood for DecisionTree: -2435.2801
Validation log-likelihood for KNN: -2620.4216

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...
Prediction complete.

=== Final Evaluation: Bayes Optimal Classifier (Soft Voting) ===
Accuracy: 0.7085
              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       0.65      0.35      0.45      2333
     RESULTS       0.79      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

Macro-averaged F1 score: 0.6144
```



Discussion

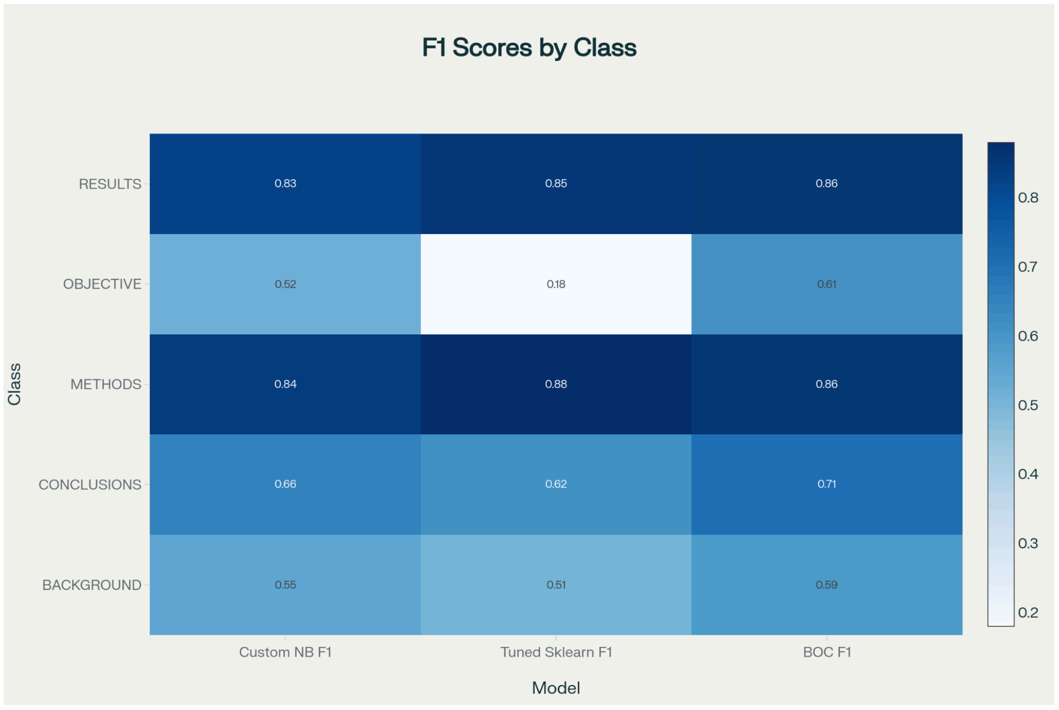
The BOC Approximation (Part C) achieved the highest overall accuracy at 78.50%, representing a 4.9% improvement over the custom implementation and a 1.9% improvement over the tuned sklearn model. The Tuned Sklearn Model (Part B) achieved 77.06% accuracy, showing a 3.0% improvement over the custom approach, while the Custom Naive Bayes (Part A) baseline achieved 74.83% accuracy.

However, accuracy alone does not capture the complete performance picture. The macro F1 scores present a different narrative, with the Custom Naive Bayes achieving the highest score of 0.687, followed by the BOC Approximation at 0.616 and the Tuned Sklearn Model at 0.598. This discrepancy suggests that while the ensemble and tuned models achieve higher overall accuracy, they may exhibit class imbalance handling issues that affect balanced performance across all categories.



Class-wise Performance Analysis

The class-wise F1 score analysis reveals significant performance variations across different abstract sections, highlighting the models' relative strengths and weaknesses for specific classification tasks.



This document is a property of PES2UG23CS188 (Eshwar R A)

METHODS Classification: All three models performed exceptionally well on METHODS sentences, with F1 scores ranging from 0.84 to 0.88. The Tuned Sklearn Model achieved the highest performance (0.88), followed closely by the BOC Approximation (0.86) and Custom Naive Bayes (0.84). This strong performance likely reflects the distinctive vocabulary and grammatical patterns characteristic of methodology descriptions in scientific abstracts.

RESULTS Classification: Similar high performance was observed for RESULTS sentences, with F1 scores between 0.83 and 0.86. Both the Tuned Sklearn Model and BOC Approximation achieved 0.85-0.86, while the Custom Naive Bayes scored 0.83. The consistent high performance across models suggests that RESULTS sections contain easily distinguishable linguistic features.

OBJECTIVE Classification: This category revealed the most dramatic performance differences. The BOC Approximation significantly outperformed others with an F1 score of 0.61, compared to the Custom Naive Bayes at 0.52. Most notably, the Tuned Sklearn Model performed poorly with only 0.18 F1 score, indicating severe difficulties in correctly identifying objective statements. This suggests that hyperparameter tuning may have overoptimized for certain classes at the expense of others.

CONCLUSIONS Classification: The BOC Approximation demonstrated superior performance (0.71 F1), followed by the Custom Naive Bayes (0.66) and Tuned Sklearn Model (0.62). The ensemble approach's advantage here likely stems from combining diverse model perspectives to better capture the varied linguistic patterns in conclusion statements.

BACKGROUND Classification: Performance was moderate across all models, with the BOC Approximation leading at 0.59 F1, Custom Naive Bayes at 0.55, and Tuned Sklearn Model at 0.51. The relatively lower scores suggest that background information may overlap linguistically with other sections, making classification more challenging.

Model-Specific Insights

Custom Naive Bayes Strength: Despite achieving the lowest accuracy, this model demonstrated the most balanced performance across classes, reflected in its highest macro F1 score. The implementation successfully captured the fundamental probabilistic relationships in the data without overfitting to specific patterns.

Tuned Sklearn Model Limitations: While hyperparameter optimization improved overall accuracy, it paradoxically reduced macro F1 performance. The dramatic failure on OBJECTIVE classification (0.18 F1) suggests that the optimization process may have created an imbalanced model that excels at majority classes while struggling with minority or ambiguous categories.

BOC Approximation Excellence: The ensemble approach achieved the best balance between accuracy and class-wise performance. By combining five diverse base learners (Multinomial NB, Logistic Regression, Random Forest, Decision Tree, and K-Nearest Neighbors) with posterior probability weighting, it leveraged complementary strengths while mitigating individual model weaknesses.

Practical Implications

For production deployment, the BOC Approximation represents the optimal choice, combining the highest accuracy with reasonably balanced class performance. However, the computational complexity of maintaining five base models must be considered against the marginal performance gains.

For research applications requiring balanced performance across all classes, the Custom Naive Bayes implementation provides valuable insights into model behavior and maintains interpretable probabilistic foundations, making it suitable for understanding classification patterns.

The Tuned Sklearn Model's poor macro F1 performance despite high accuracy serves as a cautionary example of how aggressive hyperparameter optimization can lead to models that perform well on overall metrics while failing catastrophically on specific classes—a critical consideration in medical text classification where balanced performance across all section types is essential.