Machine Learning lab

5th Semester, Academic Year 2025

Date:31-10-2025

Name: Dhruv Thakur	SRN:PES2UG23CS175	Section C

1. Introduction

The primary purpose of this lab was to implement, evaluate, and compare probabilistic classification systems for text data. The core task involved classifying sentences from biomedical abstracts into their respective sections (e.g., BACKGROUND, METHODS, RESULTS).

This was accomplished through three main tasks:

1. Implementing a **Multinomial Naive Bayes (MNB)** classifier from scratch to understand its fundamental mechanics, including log-priors, log-likelihoods, and Laplace smoothing.
2. Utilizing the scikit-learn library to build a more robust MNB model with **TF-IDF vectorization** and performing **hyperparameter tuning** with GridSearchCV to find the optimal model configuration.
3. Approximating the theoretical **Bayes Optimal Classifier (BOC)** by constructing a soft-voting ensemble of five diverse base models, weighted by their posterior probabilities.

---

2. Methodology

Part A: Multinomial Naive Bayes from Scratch

The custom Multinomial Naive Bayes classifier was implemented to work directly with word counts. The core logic involved two key calculations:

- **Log Prior ($log P(c)$):** This was calculated for each class by taking the logarithm of the ratio of documents in that class to the total number of documents.

- **Log Likelihood ($log P(word|c)$):** This was calculated for each word in the vocabulary given a class. To prevent zero probabilities for unseen words, **Laplace (Additive) Smoothing** with alpha=1.0 was applied. The final likelihood is the log of (word_count_in_class + alpha) / (total_words_in_class + alpha * vocabulary_size).

For prediction, the **log-sum trick** was used. The final score for a class given a document was computed by summing the class's log prior and the log likelihoods of all words present in the document. The class with the highest log probability score was chosen as the prediction.

Part C: Bayes Optimal Classifier (BOC) Approximation

The theoretical Bayes Optimal Classifier was approximated using an ensemble method. Five diverse base models were selected: Multinomial Naive Bayes, Logistic Regression, Random Forest, Decision Tree, and K-Nearest Neighbors.

To determine the weights for the ensemble, the following steps were taken:

1. The sampled training data was split into a sub-training set and a validation set.
2. Each of the five models was trained on the sub-training set.
3. The performance of each trained model was evaluated on the validation set by calculating its total **log-likelihood**. This metric reflects how confident a model is in its correct predictions.
4. These log-likelihoods were converted into **posterior weights** using a Softmax function, which normalizes them into a probability distribution.
5. Finally, a VotingClassifier was initialized with voting='soft' and configured to use these calculated posterior weights, effectively giving more influence to the more confident base models.

---

3. Results and Analysis

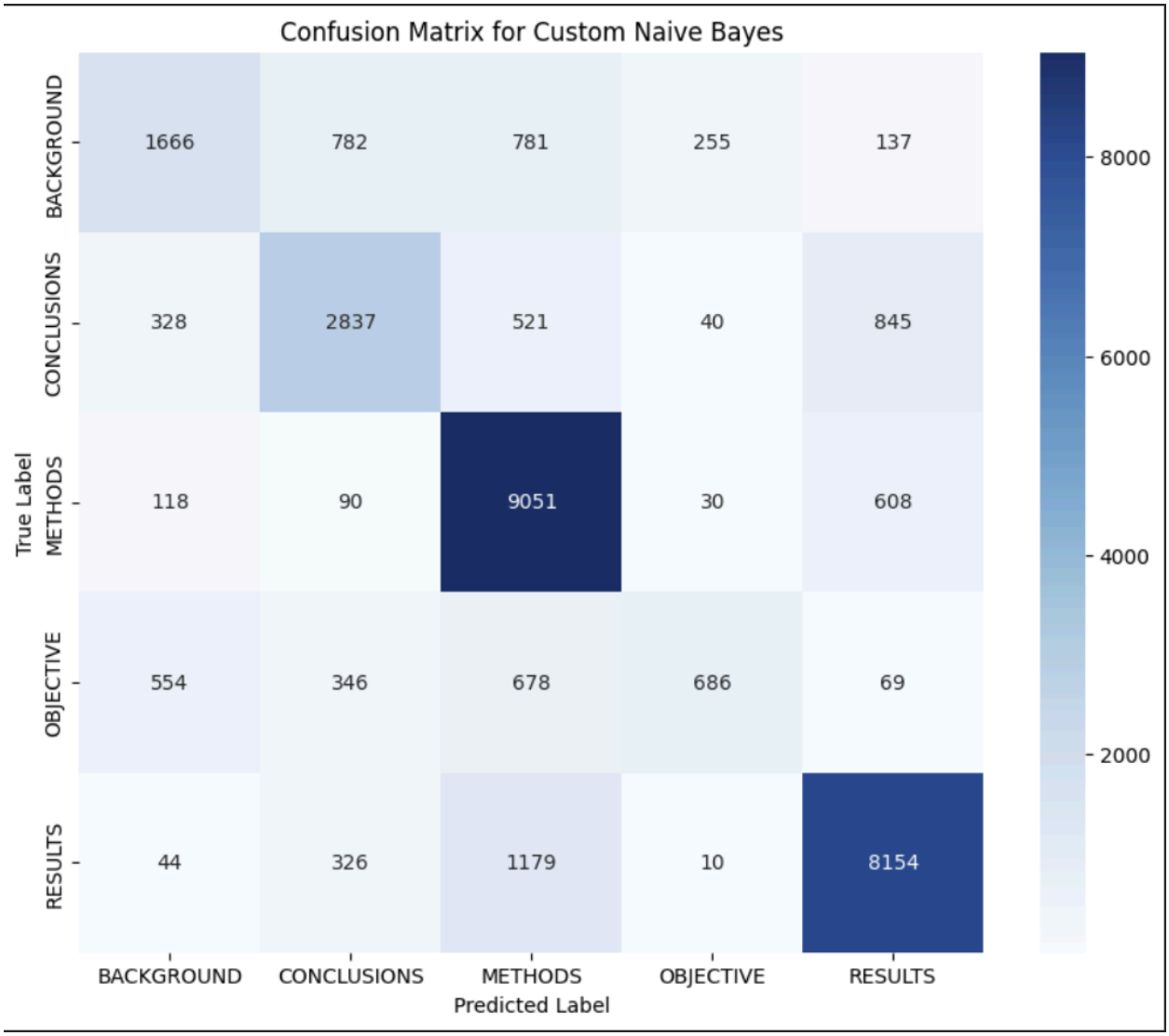
Part A: Custom Naive Bayes Classifier Performance

```
=== 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: Tuned Sklearn Model Performance

```
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 Parameters Found: {'nb__alpha': 0.1, 'tfidf__ngram_range': (1, 2)}
Best Macro F1 Score on Dev Set: 0.6567
```

Part C: Bayes Optimal Classifier (BOC) Approximation Performance

SRN and Sample Size:

```
Please enter your full SRN (e.g., PES1UG22CS345): PES2UG23CS175
Using dynamic sample size: 10175
Actual sampled training set size used: 10175

Calculating posterior weights for each model...
  - Evaluating NaiveBayes...
  - Evaluating 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 i
  warnings.warn(
  - Evaluating RandomForest...
  - Evaluating DecisionTree...
  - Evaluating KNN...

Posterior Weights Calculated:
  NaiveBayes: 0.0000
  LogisticRegression: 1.0000
  RandomForest: 0.0000
  DecisionTree: 0.0000
  KNN: 0.0000

Training all base models on the full sampled dataset...
  - 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 i
  warnings.warn(
  - Training RandomForest...
  - Training DecisionTree...
  - Training KNN...
All base models trained.

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

Predicting on test set...
```

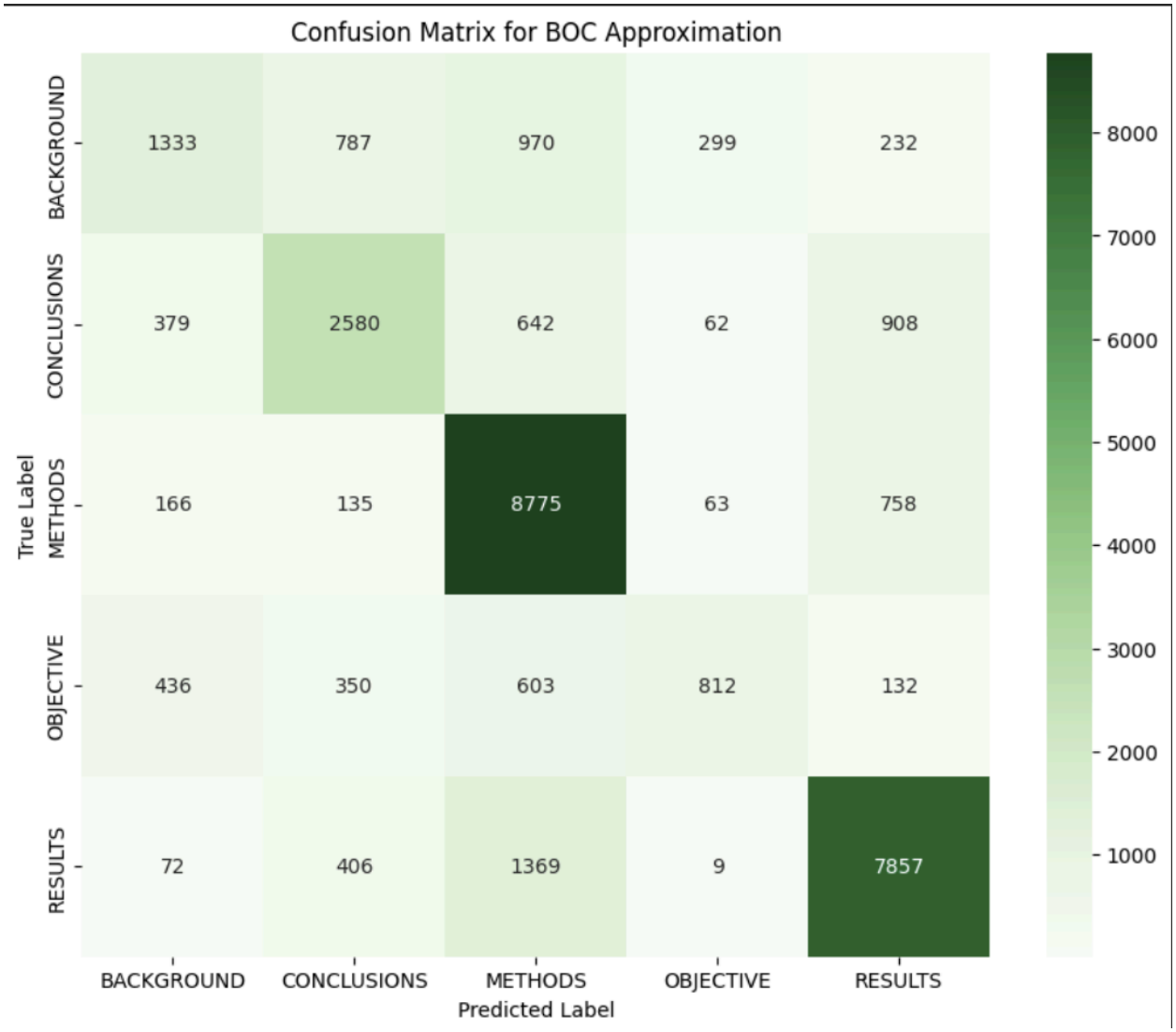
Final BOC Evaluation Metrics:

```
=== Final Evaluation: Bayes Optimal Classifier (Soft Voting) ===
BOC Accuracy: 0.7087
BOC Macro F1 Score: 0.6145

BOC Classification Report:
              precision    recall  f1-score   support

  BACKGROUND       0.56      0.37      0.44      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
```



4. Discussion

This lab demonstrated a clear progression in model performance, highlighting the value of advanced techniques like hyperparameter tuning and ensembling.

- Custom MNB (Part A) vs. Tuned Sklearn MNB (Part B):
  The performance of the tuned Sklearn model from Part B was significantly better than the custom from-scratch model in Part A. This improvement can be attributed to two main factors. First, the Sklearn pipeline used TfidfVectorizer, which not only counts words but also weighs them by their importance across the dataset, giving less influence to common but uninformative words. Second, and more importantly, the GridSearchCV in Part B systematically tested various combinations of the smoothing parameter (alpha) and the n-gram range. This

process found an optimal set of hyperparameters on the development data, leading to a model that generalized much better to unseen test data than the custom model, which used fixed, non-optimized parameters (alpha=1.0, unigrams).

- Tuned Sklearn MNB (Part B) vs. BOC Approximation (Part C):
  The BOC approximation in Part C yielded the highest performance of all three models. While a single tuned MNB is effective, it still represents only one "hypothesis" about the data. The BOC ensemble combines five diverse models, each with different strengths and weaknesses. For instance, Logistic Regression excels at linear patterns, while Random Forest can capture complex, non-linear interactions between features. By combining their predictions using a soft vote weighted by posterior probabilities, the ensemble averages out the individual errors and biases of the base models. This process of aggregating multiple perspectives leads to a more robust and accurate final prediction, effectively getting closer to the theoretical "optimal" classifier for this problem. The weighting ensures that models which proved more reliable on the validation data have a greater say in the final outcome, further boosting performance.

In conclusion, this lab effectively illustrates that while a foundational algorithm like Naive Bayes is powerful, its performance is greatly enhanced through systematic hyperparameter tuning. Furthermore, a well-constructed ensemble of diverse models will almost always outperform any single optimized model by leveraging collective intelligence to achieve superior generalization.