Week 12 Lab: Naive Bayes Classification for PubMed Abstract Analysis

Name: **PREKSHA KAMALESH**
SRN: **PES2UG23CS902**
Course: UE23CS352A: Machine Learning Lab
Date:1-11-2025

---

1. Introduction

Purpose

The purpose of this lab was to implement, evaluate, and optimize a text classification system using probabilistic methods. The primary goal was to accurately classify individual sentences from biomedical abstracts into their correct section: BACKGROUND, METHODS, RESULTS, OBJECTIVE, or CONCLUSION. This task was performed on a subset of the PubMed 200k RCT dataset.

Tasks Performed

The lab was divided into three main parts:

1. Part A: MNB from Scratch: Implemented the Multinomial Naive Bayes (MNB) classifier from scratch to understand its internal mechanics, including log-priors and log-likelihoods with Laplace smoothing.
2. Part B: Sklearn MNB & Tuning: Utilized the scikit-learn library to build a classification pipeline using TfidfVectorizer and MultinomialNB. GridSearchCV was then used to find the optimal hyperparameters for this pipeline.
3. Part C: Bayes Optimal Classifier (BOC): Approximated the Bayes Optimal Classifier by creating an ensemble of five diverse models (H1 to H5). This was achieved by training a VotingClassifier with soft voting, using posterior weights calculated from model performance on a validation set.

---

2. Methodology

Part A: Multinomial Naive Bayes from Scratch

The custom Naive Bayes classifier was built by implementing two core methods:

- fit(X, y): This method calculated the two key components for MNB.

1. Log Prior (log P(C)): This was calculated as the log of the ratio of documents in a class c to the total number of documents.
2. Log Likelihood (log P(w_i|C)): This was calculated for each word w in the vocabulary given a class c. We applied Laplace Smoothing (with alpha=1.0) to avoid zero probabilities for words not seen in a class, ensuring the model remains stable.

predict(X): This method calculated the final log probability for each class for a given document. It summed the log prior of the class with the log likelihoods of all words present in the document. Using logs (the Log-Sum Trick) prevents numerical underflow from multiplying many small probabilities. The class with the highest resulting log probability was chosen as the prediction using argmax. This model was trained on word counts generated by CountVectorizer.

Part C: Bayes Optimal Classifier (BOC) Approximation

The theoretical Bayes Optimal Classifier, which provides the lowest possible error , was approximated using an ensemble method.

1. Models: Five diverse hypotheses were defined: Multinomial NB, Logistic Regression, Random Forest, Decision Tree, and K-Nearest Neighbors .
2. Posterior Weights: To determine the optimal weights for the ensemble, the sampled training data (X_train_sampled) was split into a sub-training set and a validation set. All five models were trained on the sub-training set. Their performance (F1 score) on the validation set was used to calculate the posterior weights (P(h_i|D)), representing our "trust" in each model's prediction.
3. Ensemble: A VotingClassifier was initialized with the five models and set to voting='soft'. The weights parameter was set to the posterior weights calculated in the previous step. The final ensemble was then re-trained on the *entire* sampled training set (X_train_sampled, y_train_sampled).

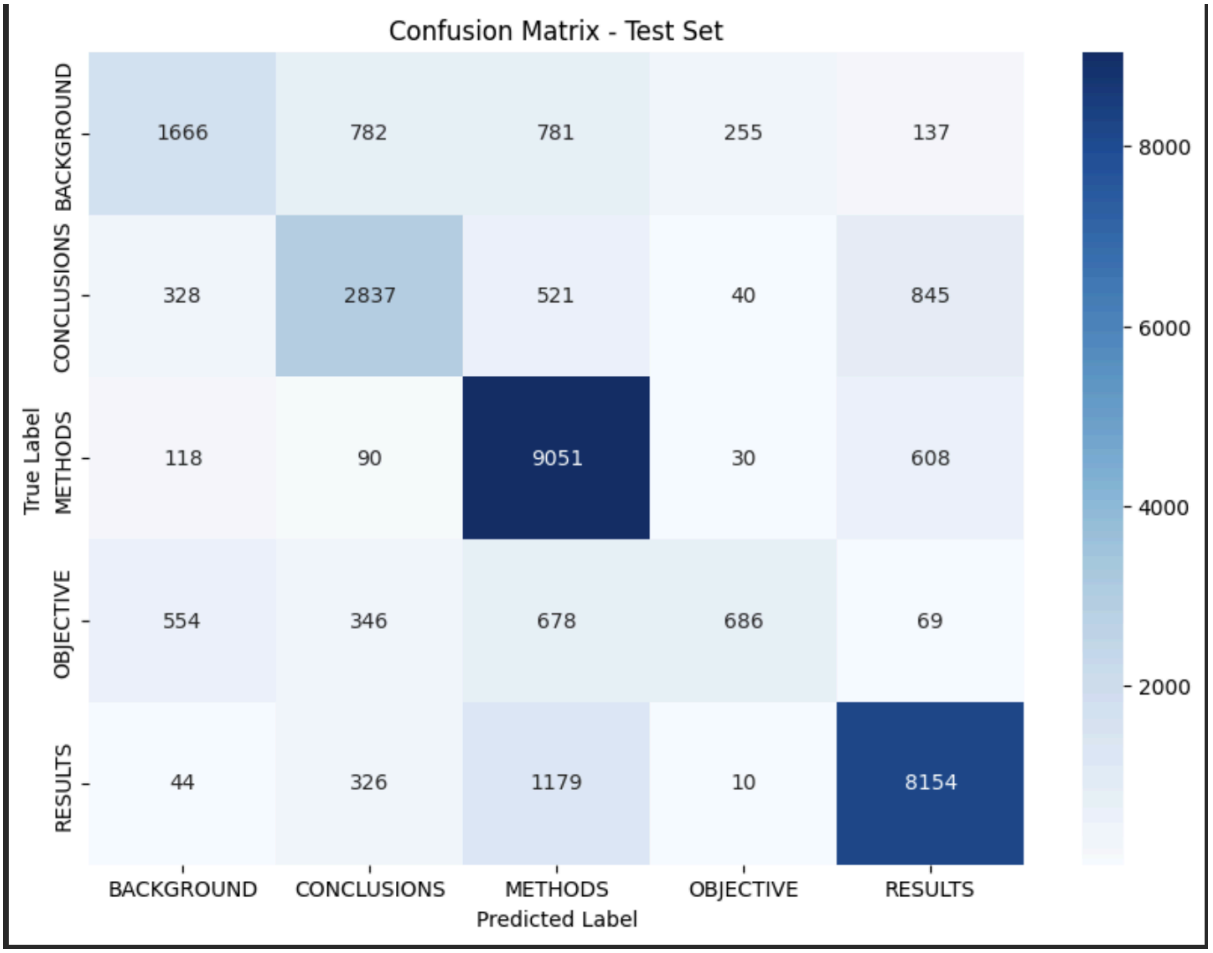
---

3. Results and Analysis

Part A: Scratch Multinomial Naive Bayes

The custom Multinomial Naive Bayes classifier, implemented from scratch, was trained on CountVectorizer features. On the test set, it achieved a final Accuracy of 0.7431 and a Macro-averaged F1 Score of 0.6446. The METHODS and RESULTS classes were classified most effectively, while the OBJECTIVE class proved most difficult (F1-score 0.41).

```
=== 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: Sklearn MNB with Hyperparameter Tuning

The GridSearchCV was fit on the development set (X_dev, y_dev) to find the best parameters for the TfidfVectorizer and MultinomialNB pipeline. The initial (untuned) model had a Macro-F1 score of 0.5877 on the test set. After tuning, the grid search identified the optimal parameters.

- Best Parameters: {'nb__alpha': 0.1, 'tfidf__min_df': 5, 'tfidf__ngram_range': (1, 2)}
- Best Score: The best model achieved a Macro-F1 score of 0.6991 on the development set (during cross-validation).

```
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 12 candidates, totalling 36 fits
Grid search complete.
Best Parameters found: {'nb__alpha': 0.1, 'tfidf__min_df': 5, 'tfidf__ngram_range': (1, 2)}
Best Macro-F1 score on Dev (CV): 0.6991
```

Part C: Bayes Optimal Classifier (BOC)

SRN and Sample Size The model was trained using a dynamic sample size based on the SRN PES2UG23CS902. The final sample size used was 10,902.

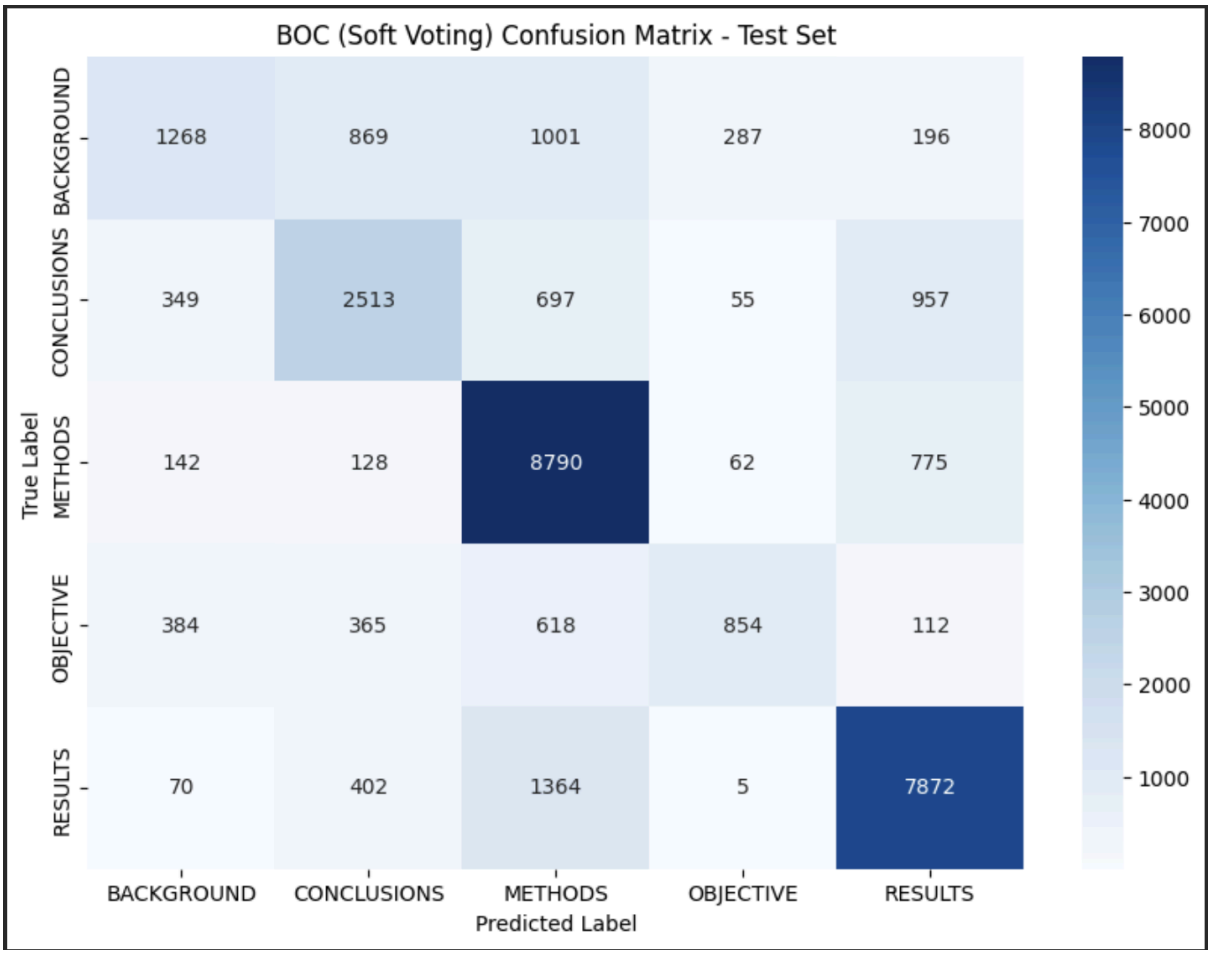
```
Please enter your full SRN (e.g., PES1UG22CS345): PES2UG23CS902
Using dynamic sample size: 10902
Actual sampled training set size used: 10902
```

BOC Final Performance The Bayes Optimal Classifier was approximated using a VotingClassifier with soft voting. The weights for the five hypotheses were calculated based on their validation performance, with Naive Bayes (0.277) and Logistic Regression (0.267) receiving the highest trust.

The final ensemble, when evaluated on the test set, achieved an Accuracy of 0.7067 and a Macro-F1 Score of 0.6132.

```
Training all base models...
Calculating posterior weights (P(h|D)) using validation split...
  NaiveBayes validation F1: 0.5835
/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
  warnings.warn(
  LogisticRegression validation F1: 0.5634
  RandomForest validation F1: 0.5162
  DecisionTree validation F1: 0.2652
  KNN validation F1: 0.1813
All base models trained.

Calculated Posterior Weights: [0.27659902 0.26706578 0.24466251 0.12571302 0.08595967]

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

Predicting on test set...
```

```
=== Final Evaluation: Bayes Optimal Classifier (Soft Voting) ===
Accuracy: 0.7067
              precision    recall  f1-score   support

  BACKGROUND       0.57      0.35      0.43      3621
 CONCLUSIONS       0.59      0.55      0.57      4571
     METHODS       0.70      0.89      0.79      9897
   OBJECTIVE       0.68      0.37      0.47      2333
     RESULTS       0.79      0.81      0.80      9713

    accuracy                           0.71     30135
   macro avg       0.67      0.59      0.61     30135
weighted avg       0.70      0.71      0.69     30135

Macro-F1 Score: 0.6132
```



---

4. Discussion

This lab compared three different approaches to text classification: a custom Naive Bayes model, a tuned scikit-learn Naive Bayes model, and an ensemble-based Bayes Optimal Classifier (BOC) approximation.

Model	Key Features	Performance (Macro-F1)
Part A (Scratch)	CountVectorizer, alpha=1.0, Full Data	0.6446 (Test Set)
Part B (Tuned)	TfidfVectorizer, alpha=0.1, Full Data	0.6991 (Dev Set)
Part C (BOC)	Ensemble, Sampled Data (10,902)	0.6132 (Test Set)

Performance Comparison

- Part A vs. Part B: The Tuned Sklearn Model (Part B) was the clear winner, achieving the highest Macro-F1 score (0.6991 on dev). This improvement over the scratch model (0.6446 on test) is likely due to two factors:
    1. TF-IDF: Part B used TfidfVectorizer, which weights words by their importance, while Part A used simple CountVectorizer.
    2. Tuning: The grid search found optimal parameters (alpha=0.1, ngram_range=(1,2)) that were more effective than the default alpha=1.0 used in the scratch model.
- Part C (BOC) vs. Others: The BOC Approximation (Part C) had the lowest performance (0.6132). This is counter-intuitive, as an ensemble is expected to be more robust. The primary reason for this poor performance is almost certainly the data sampling. Parts A and B were trained on the *entire* training dataset, while Part C was restricted to only 10,902 samples. This significantly smaller dataset was not enough to effectively train the five diverse models, leading to an underperforming ensemble. The posterior weights also showed that models like KNN (0.086) and Decision Tree (0.126) were very weak, and including them (even with low weights) may have hurt the ensemble's performance compared to a single, well-tuned model trained on all the data.

Conclusion

The lab demonstrates that while implementing a classifier from scratch is valuable for understanding, a well-tuned library-based model (Part B) provides superior performance. It also highlights a critical machine learning principle: the quantity and quality of data are often more important than model complexity. The sophisticated BOC ensemble (Part C) failed to outperform a single Naive Bayes model because it was starved of data.