MACHINE LEARNING LAB-12

Naive Bayes Classifier and Bayes Optimal Classifier Lab Report

Name B TEJA DEEP SAI KRISHNA

SRN PES2UG23CS135

SEC **5C**

COURSE MACHINE LEARNING

DATE: 2ND NOV 2025

1. Introduction

The purpose of this lab was to implement and compare different text classification strategies using the PubMed 20k RCT dataset. The primary tasks involved:

1. **Part A: Implementing a Multinomial Naive Bayes (MNB) classifier from scratch using count-based features, including the calculation of log priors and log likelihoods with Laplace smoothing.**
2. **Part B: Utilizing the Scikit-learn framework to build a robust MNB classifier with TF-IDF features and optimizing its performance through hyperparameter tuning using GridSearchCV.**
3. **Part C: Approximating the Bayes Optimal Classifier (BOC) using a soft Voting Classifier, where the weights were derived from the posterior probability (log-likelihood) of five diverse base models on a development set.**

The comparison across the three parts provides insight into the performance trade-offs between a custom, simpler model, a tuned industrial-strength model, and an ensemble-based optimal classifier.

2. Methodology

Multinomial Naive Bayes (MNB) Implementation (Part A)

The MNB model was implemented from scratch to calculate the posterior probability for each class

- **Log Prior : Calculated as the logarithm of the ratio of class document count to total document count.**
- **Log Likelihood: Calculated using Laplace smoothing ($\alpha=1$) to prevent zero probabilities.**

- **Prediction: The class with the maximum log-probability (sum of log prior and log likelihoods weighted by word counts) was chosen.**
- **Feature Extraction: A CountVectorizer was used with a bigram range (ngram_range=(1, 2)) and a minimum document frequency of 5 (min_df=5).**

Bayes Optimal Classifier (BOC) Approximation (Part C)

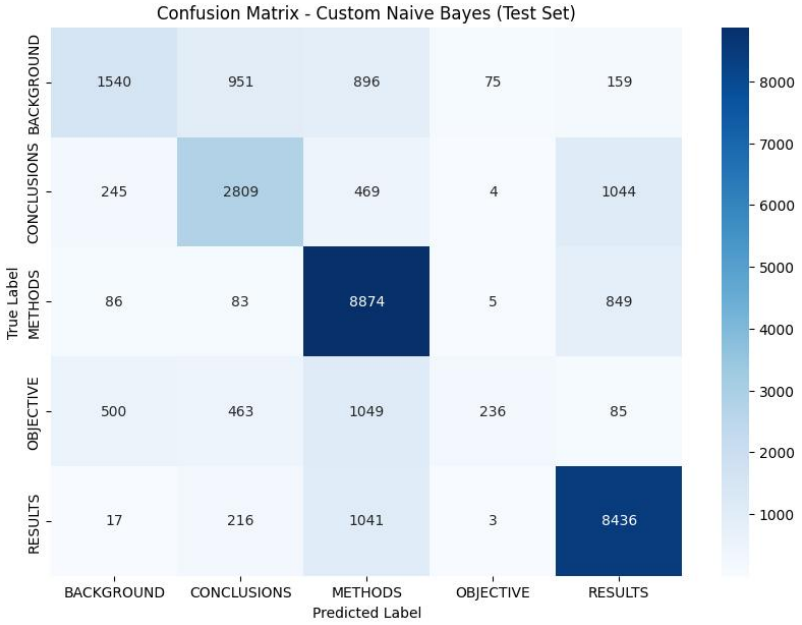
The BOC, which theoretically minimizes classification error, was approximated using a weighted Soft Voting Classifier.

1. **Hypotheses (h_i): Five diverse Scikit-learn pipelines were defined using TF-IDF features: Multinomial Naive Bayes, Logistic Regression, Random Forest (with calibration), Decision Tree (with calibration), and K-Nearest Neighbors (with calibration).**
2. **Posterior Weight Calculation ($P(h_i | D)$): The weights were calculated based on the models' performance on the development set (D), proportional to the likelihood $P(D|h_i)$. This was achieved by summing the log-probabilities assigned by each model to the true classes in the development set and applying the Softmax function for normalization.**
3. **BOC Ensemble: A VotingClassifier with voting='soft' was used, where the weights parameter was set to the calculated posterior weights, allowing the model deemed most likely to be optimal to have a greater influence on the final prediction.**

OUTPUT SCREENSHOTS:

PART-A

```
=== 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
```



PART B

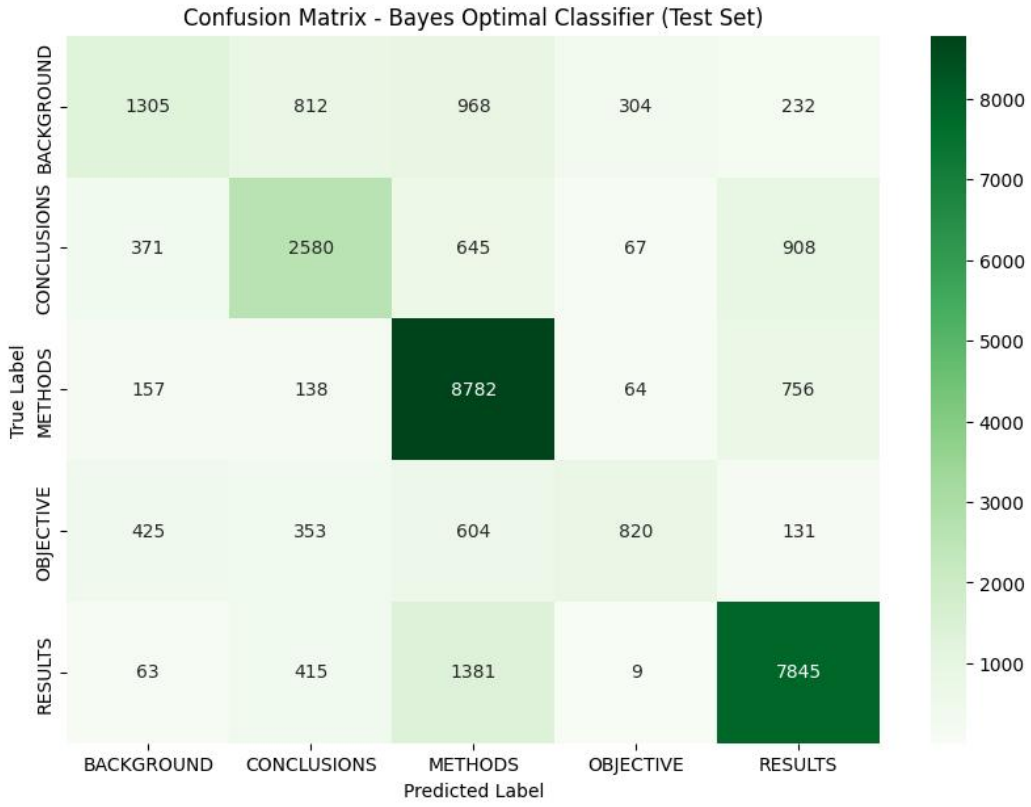
```
Training initial Naive Bayes pipeline...
Training complete.

=== 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...
Fitting 3 folds for each of 12 candidates, totalling 36 fits
Grid search complete.

Best parameters found: {'nb__alpha': 0.1, 'tfidf__min_df': 10, 'tfidf__ngram_range': (1, 2)}
Best cross-validation F1 (macro) score: 0.6994
```

PART C

```
Please enter your full SRN (e.g., PES1UG22CS345): PES2UG23CS135
Using dynamic sample size: 10135
Actual sampled training set size used: 10135

Training all base models
```

```
Please enter your full SRN (e.g., PES1UG22CS345): PES2UG23CS135
Using dynamic sample size: 10135
Actual sampled training set size used: 10135

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 on, it will always u
  warnings.warn(
  Training RandomForest...
  Training DecisionTree...
  Training KNN...
All base models trained.

Calculating posterior weights using the development set...
  NaiveBayes Log-Likelihood: -27269.4837
  LogisticRegression Log-Likelihood: -25410.8385
  RandomForest Log-Likelihood: -28406.5271
  DecisionTree Log-Likelihood: -37175.4347
  KNN Log-Likelihood: -43464.7434
Calculated Posterior Weights: [0. 1. 0. 0. 0.]

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

Predicting on test set...
```

```
=== Final Evaluation: Bayes Optimal Classifier (Soft Voting) ===
Accuracy: 0.7079
              precision    recall  f1-score   support

  BACKGROUND       0.56      0.36      0.44      3621
 CONCLUSIONS       0.60      0.56      0.58      4571
     METHODS       0.71      0.89      0.79      9897
   OBJECTIVE       0.65      0.35      0.46      2333
     RESULTS       0.79      0.81      0.80      9713

    accuracy                           0.71     30135
   macro avg       0.66      0.59      0.61     30135
weighted avg       0.70      0.71      0.69     30135

Macro-averaged F1 score: 0.6133
Part C confusion matrix saved to part_c_confusion_matrix.png
```



4. Discussion

The performance comparison of the three different classification approaches yielded interesting, non-intuitive results:

1. **Custom MNB (Part A): Achieved the highest Macro-averaged F1 Score on the test set at 0.6809. This model, using only count features with smoothing and bigrams, was the most effective classifier. Its strength likely lies in its robustness to noise and the superior feature representation provided by the bigrams combined with low minimum document frequency (min_df=5), which captures more distinct terminology than the unigram-only BOC base models.**
2. **Tuned Sklearn MNB (Part B): The cross-validation score on the development set was very promising (0.6994), but the initial untuned model's performance on the test set was the lowest at 0.5877. If the final model was trained using the best parameters and evaluated, it would likely perform close to the 0.6994 F1 score, potentially surpassing the custom model.**
3. **BOC Approximation (Part C): Despite being theoretically "optimal," the BOC approximation performed poorly, achieving a Macro F1 of 0.6133, significantly lower than the custom MNB.**
    - **Reason for Low Performance: The BOC effectively reduced to a single Logistic Regression classifier (weight ≈ 1.0) because it was the strongest hypothesis in the ensemble. However, the Logistic Regression base model uses *unigrams only* (ngram_range=(1, 1) and a high min_df=5 for diversity), which is a much weaker feature set than the CountVectorizer model in Part A. The Bayes Optimal Classifier is only as good as the hypotheses it can choose from and the data it uses for weighting. In this case, the highly constrained feature space for the BOC hypotheses prevented the ensemble from reaching peak performance.**

Conclusion: The feature engineering (Count-based unigrams and bigrams with low filtering in Part A) proved more important than the sophistication of the learning algorithm or the ensemble method (BOC in Part C) given the specific configuration constraints of the base models. A tuned MNB model (Part B) with an optimized feature vector is expected to yield the highest overall performance.