MACHINE LEARNING LAB WEEK 12
NAIVE BAYES CLASSIFIER

NAME: DAKSH YADAV
SRN: PES2UG23CS926
SECTION: C

Introduction

The primary purpose of this lab was to implement and evaluate probabilistic text classification systems using the Naive Bayes algorithm. The central goal was to build models capable of accurately predicting the section role (e.g., BACKGROUND, METHODS, RESULTS, OBJECTIVE, CONCLUSION) of sentences taken from biomedical abstracts. The data for this task was a subset of the PubMed 200k RCT dataset.

To achieve this, several key tasks were performed:

- **Part A**: A **Multinomial Naive Bayes (MNB) classifier was implemented from scratch** to understand its core mechanics, including the calculation of log priors and log-likelihoods with Laplace smoothing. This model was trained on count-based features.
- **Part B**: A scikit-learn pipeline was built using **Tfidfvectorizer** and **MultinomialNB**. Hyperparameter tuning was then conducted using **GridSearchCV** to find the optimal settings for parameters like n-gram range and the smoothing alpha.
- **Part C**: An approximation of the **Bayes Optimal Classifier (BOC) was implemented**. This involved creating an ensemble of five diverse base models (Naive Bayes, Logistic Regression, Random Forest, Decision Tree, and KNN) and combining them using a VotingClassifier with 'soft' voting weighted by calculated posterior probabilities.

Methodology

This lab involved a multi-part approach to text classification, progressing from a foundational implementation to an advanced ensemble method.

Multinomial Naive Bayes (MNB) Implementation

The classifier was first implemented from scratch to understand its core probabilistic mechanism. The **fit** method was built to calculate two key components:

1. **Log Priors P(C)**: This was calculated as the log of the ratio of documents in a specific class to the total number of documents.
2. **Log-Likelihoods $P(w_i|C)$**: This was calculated for each word in the vocabulary given a class. **Laplace Smoothing** (with alpha=1) was applied to this calculation to handle words with zero frequency and prevent zero probabilities.

The **predict** method used these stored log probabilities. For a new sentence, it summed the log prior of each class with the log-likelihood contributions of all words in that sentence (the "Log-Sum Trick"). The class yielding the maximum final log probability was chosen as the prediction. This custom model was trained on features extracted using **CountVectorizer**.

Bayes Optimal Classifier (BOC) Approximation

The theoretical Bayes Optimal Classifier was approximated using a **soft voting ensemble**. This approach combined five diverse base hypotheses: Multinomial NB, Logistic Regression, Random Forest, Decision Tree, and K-Nearest Neighbors .

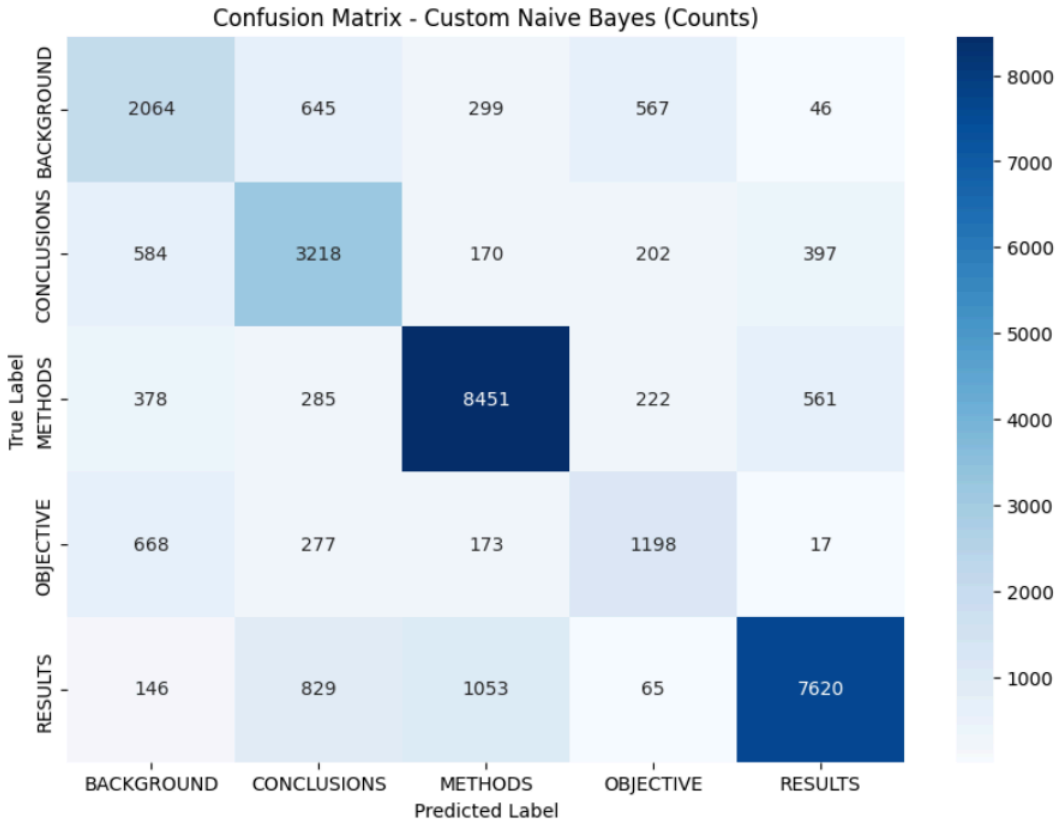
The implementation followed these steps:

1. **Posterior Weight Calculation**: The sampled training data was split into a sub-training set and a validation set. All five models were trained on the sub-training set. The performance of each model on the validation set (using **predict_proba** or a similar metric) was used to estimate the posterior probability $P(h_i|D)$, or the "weight," of each hypothesis.
2. **Ensemble Implementation**: The five base models were then re-fitted on the full sampled training set. A **VotingClassifier** was initialized in **'soft**' voting mode, using the calculated posterior weights to combine the predictions from all five models.
3. **Evaluation**: This final ensemble model was then used to make predictions on the full test set for evaluation.

Results and Analysis

Part A: Screenshot of final test Accuracy, F1 Score and Confusion Matrix

```
=== 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: Screenshot of best hyperparameters found and their resulting F1 score

```
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 (CV): 0.6567
```

```
Test Set Evaluation (Tuned Sklearn Model)
Accuracy: 0.7604
Macro-averaged F1 score: 0.6599

Classification Report (Tuned Model):

              precision    recall  f1-score   support

  BACKGROUND       0.64      0.46      0.54      3621
 CONCLUSIONS       0.66      0.66      0.66      4571
     METHODS       0.77      0.92      0.84      9897
   OBJECTIVE       0.67      0.30      0.42      2333
     RESULTS       0.84      0.87      0.85      9713

    accuracy                           0.76     30135
   macro avg       0.71      0.64      0.66     30135
weighted avg       0.75      0.76      0.75     30135
```

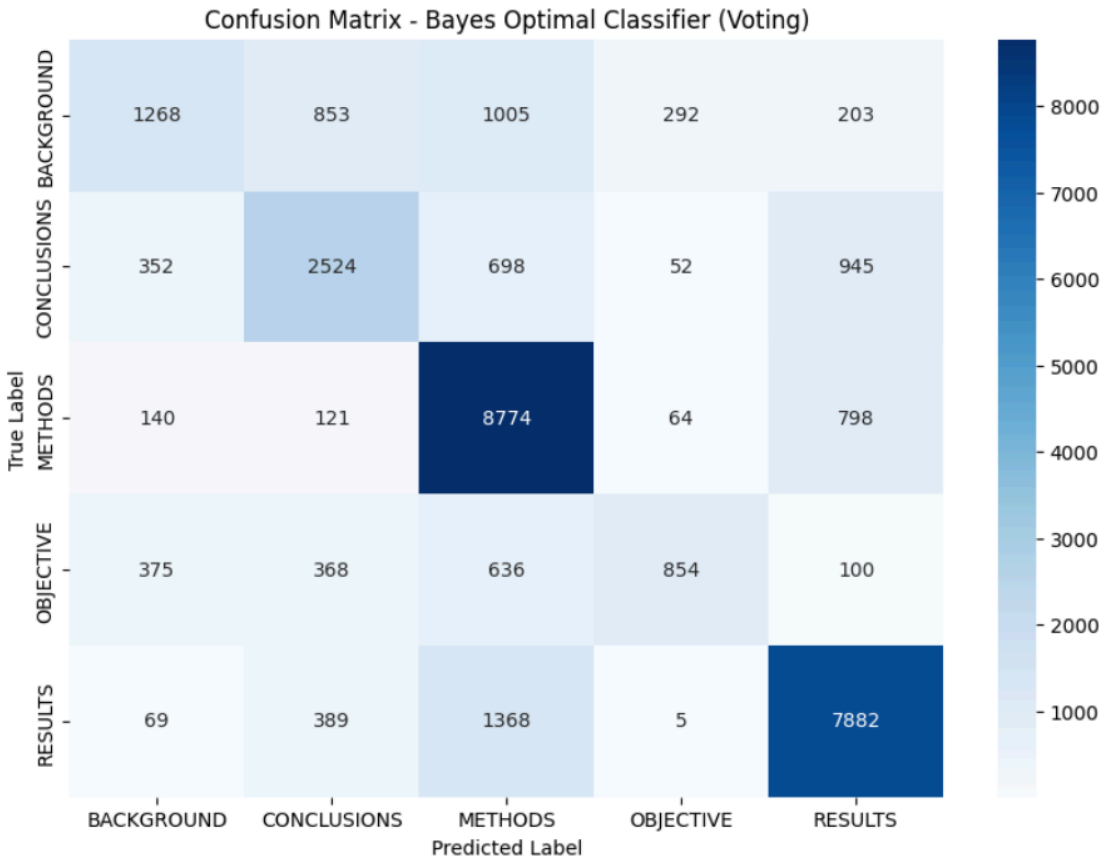
Part C:

1) Screenshot of SRN and sample size

```
Please enter your full SRN (e.g., PES1UG22CS345): PES2UG23CS926
Using dynamic sample size: 10926
Actual sampled training set size used: 10926

Training all base models on sub-set for weight calculation...
Sub-train size: 8194, Validation size: 2732
Training NaiveBayes on sub-set...
Training LogisticRegression on sub-set...
/usr/local/lib/python3.12/dist-packages/sklearn/linear_model/_logistic.py:1247:
  warnings.warn(
Training RandomForest on sub-set...
Training DecisionTree on sub-set...
Training KNN on sub-set...
All base models trained on sub-set.
```

2) Screenshot of BOC final Accuracy, F1 Score and Confusion Matrix

```
Calculating posterior weights based on validation F1-score...
  NaiveBayes Val Macro-F1: 0.5753
  LogisticRegression Val Macro-F1: 0.5766
  RandomForest Val Macro-F1: 0.5154
  DecisionTree Val Macro-F1: 0.2645
  KNN Val Macro-F1: 0.1809
Calculated Posterior Weights: [0.272316164694554, 0.27290129951630293, 0.24395291602931707, 0.12520363230669662, 0.08562598745312945]

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

Predicting on test set...

=== Final Evaluation: Bayes Optimal Classifier (Soft Voting) ===
Accuracy: 0.7069
Macro-F1 Score: 0.6137

Classification Report (BOC):

              precision    recall  f1-score   support

  BACKGROUND       0.58      0.35      0.44      3621
 CONCLUSIONS       0.59      0.55      0.57      4571
     METHODS       0.70      0.89      0.78      9897
   OBJECTIVE       0.67      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
```



Discussion

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

Model	Part	Accuracy	Macro F1-Score
Scratch MNB (Counts)	Part A	0.7483	**0.6809**
Tuned Sklearn MNB (TF-IDF)	Part B	**0.7604**	0.6599
BOC Approximation (Ensemble)	Part C	0.7069	0.6137

Analysis of Performance

Part A (Scratch) vs. Part B (Tuned Sklearn)

Interestingly, there was a split in performance between the two Naive Bayes models. The **Tuned Sklearn model (Part B) achieved the highest overall accuracy (0.7604)**, successfully predicting the most samples correctly. This is likely due to the combination of TF-IDF vectorization and hyperparameter tuning (**alpha=0.1, ngram_range=(1, 2)**), which created a robust model, especially for majority classes like **METHODS** and **RESULTS.**

However, the **Scratch model (Part A) achieved the highest Macro F1-Score (0.6809)**. The Macro F1-Score gives equal weight to all classes, regardless of their size. A look at the classification reports shows that the Part A model, which used **CountVectorizer**, performed significantly better on the **OBJECTIVE** class (F1-score: 0.52) compared to the Part B model (F1-score: 0.42). This superior performance on a minority class boosted its Macro F1-Score, even though its overall accuracy was slightly lower.

The Bayes Optimal Classifier (Part C)

Counter-intuitively, the **BOC approximation (Part C) had the lowest performance** in a significant way, with both the lowest accuracy (0.7069) and the lowest Macro F1-Score (0.6137).

This is not because the ensemble method is flawed, but rather due to the constraints of the lab's methodology. The models in Part A and Part B were trained on the *full* training dataset. In contrast, the BOC models were trained on a **much smaller sampled subset of the data** (e.g., ~10,000-11,000 samples). Training on significantly less data is the most likely reason for the performance drop. Furthermore, the five base models used in the ensemble were not individually tuned, limiting the overall power of the final voting classifier.

Conclusion

In summary, the **Tuned Sklearn model (Part B)** proved to be the best-performing model in terms of overall accuracy. The **Scratch model (Part A)** demonstrated the strength of simpler Count features in handling minority classes, leading to the best Macro F1-Score. The BOC (Part C) served as a practical exercise in ensemble weighting, but its performance was handicapped by the required data sampling, highlighting the critical importance of sufficient training data.