Machine Learning ~ Lab12

Name : Nagula Anish

SRN : PES2UG23CS358

Section : F

Course : UE23CS352A

Date: October 30, 2025

1. Introduction

The goal of this lab was to understand and use Naive Bayes for text classification. The task was to take sentences from medical abstracts and classify them into their correct section, like "METHODS" or "RESULTS". I had to implement the classifier from scratch and then use `sklearn` to build a better one and finally create an ensemble model to approximate the Bayes Optimal Classifier.

2. Methodology

a. ***MNB***

First, I had to build my own Multinomial Naive Bayes classifier. I implemented the `fit` method to calculate the log prior $(logP(C))$for each class and the log likelihood $(logP(wi \mid C))$ for each word. I made sure to include Laplace smoothing to avoid zero probabilities which would have broken the calculations. My `predict` method then added up the log scores and picked the class with the highest score.

Next, I used `sklearn`'s tools, which was much faster. I built a `Pipeline` that combined the `TfidfVectorizer` and `MultinomialNB` classifier. To find the best settings, I used `GridSearchCV` on the development data (`dev.txt`). I specifically tuned the `ngram_range` and the `alpha` smoothing parameter.

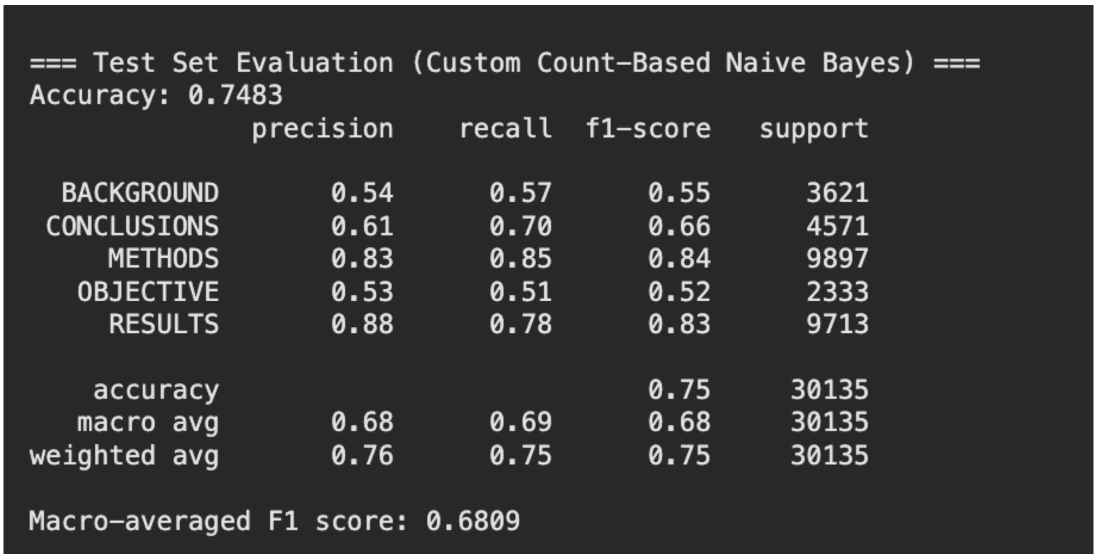
b. ***BOC***

For the last part, I tried to approximate the Bayes Optimal Classifier. I used a `VotingClassifier` that combined five different models: Naive Bayes, Logistic Regression, Random Forest, Decision Tree, and KNN . The key step was calculating posterior weights for each model by training them on a sub-split of the data and seeing how well they performed on a validation set. I then used these weights in the `VotingClassifier` with `voting='soft'` .

---

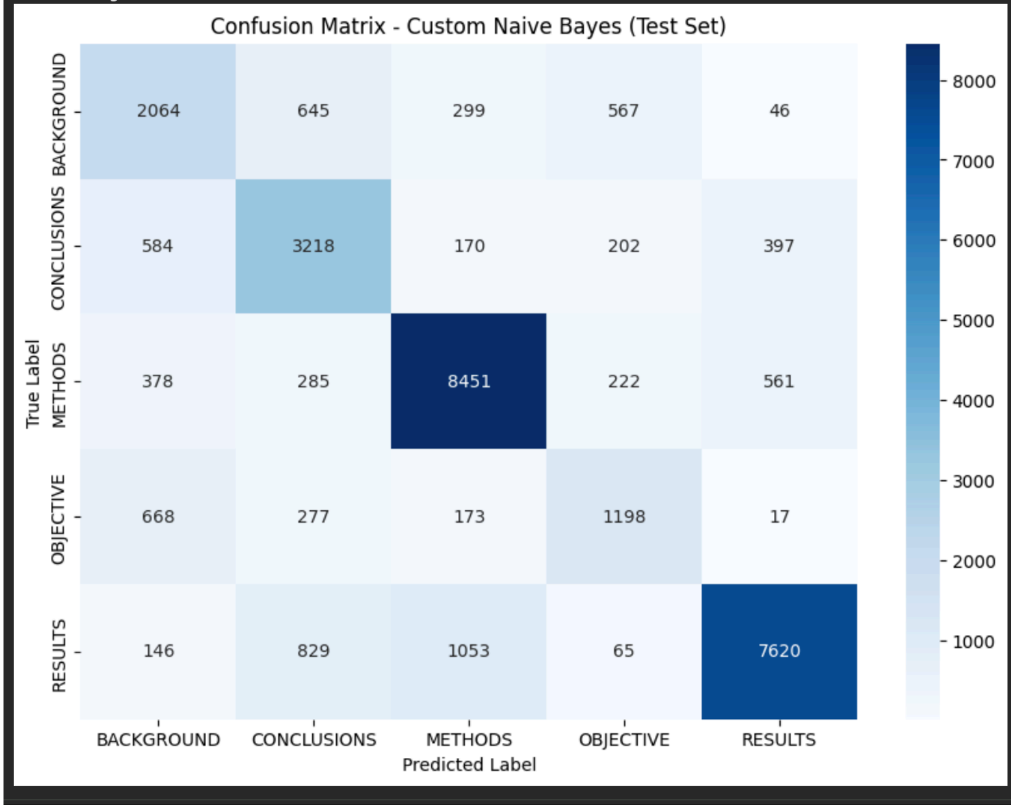
3. Results & Analysis

a. ***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
```

```
Generating Confusion Matrix...
```

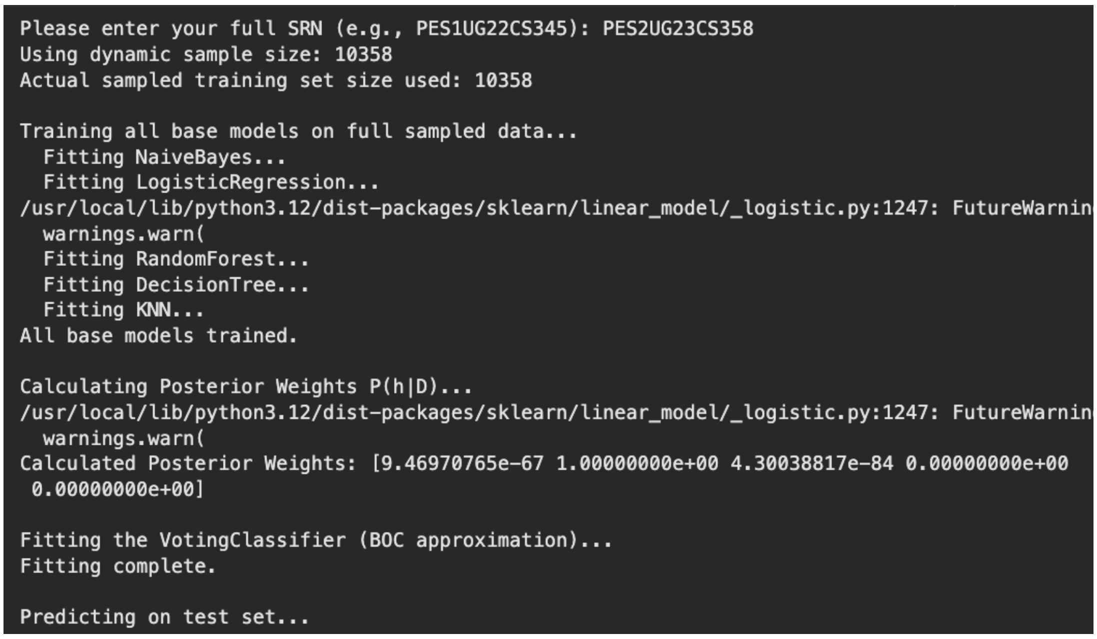


True \ Predicted	BACKGROUND	CONCLUSIONS	METHODS	OBJECTIVE	RESULTS
BACKGROUND	2064	645	299	567	46
CONCLUSIONS	584	3218	170	202	397
METHODS	378	285	8451	222	561
OBJECTIVE	668	277	173	1198	17
RESULTS	146	829	1053	65	7620

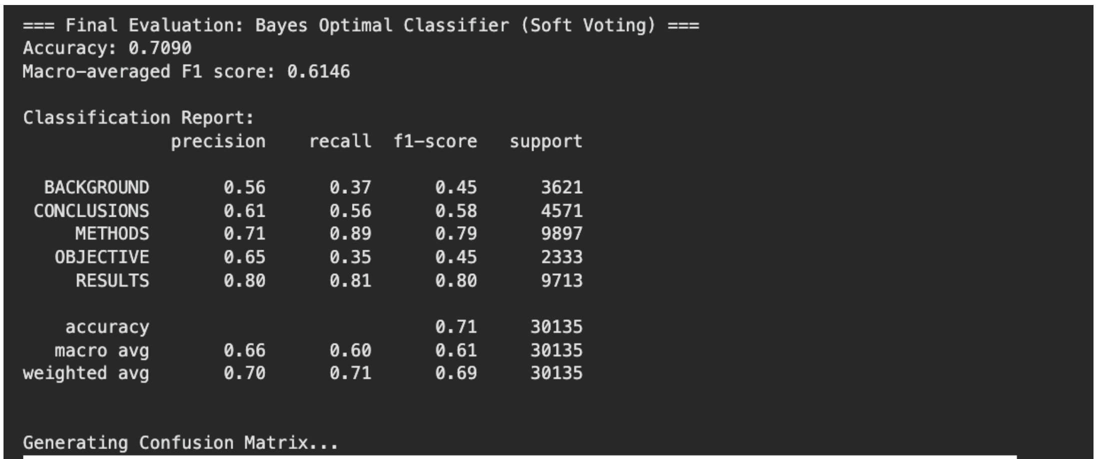
b. ***PART~B***

```
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 6 candidates, totalling 18 fits
Grid search complete.

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

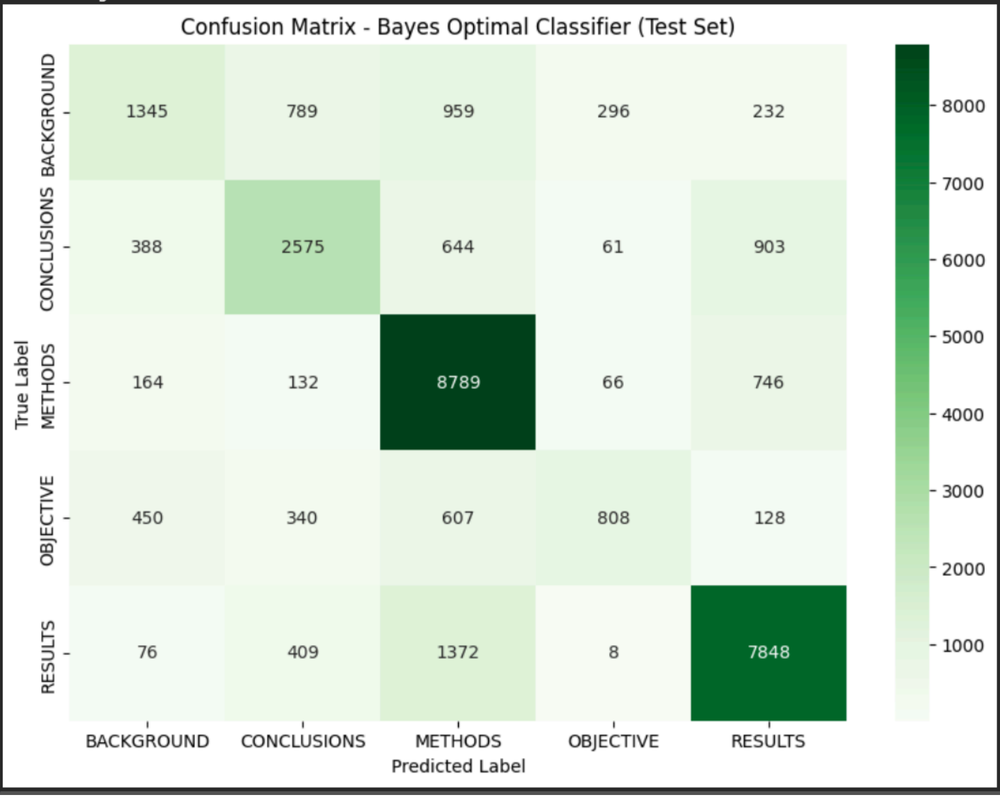
c. ***PART~C***



```
Please enter your full SRN (e.g., PES1UG22CS345): PES2UG23CS358
Using dynamic sample size: 10358
Actual sampled training set size used: 10358

Training all base models on full sampled data...
  Fitting NaiveBayes...
  Fitting LogisticRegression...
/usr/local/lib/python3.12/dist-packages/sklearn/linear_model/_logistic.py:1247: FutureWarnin
  warnings.warn(
  Fitting RandomForest...
  Fitting DecisionTree...
  Fitting KNN...
All base models trained.

Calculating Posterior Weights P(h|D)...
/usr/local/lib/python3.12/dist-packages/sklearn/linear_model/_logistic.py:1247: FutureWarnin
  warnings.warn(
Calculated Posterior Weights: [9.46970765e-67 1.00000000e+00 4.30038817e-84 0.00000000e+00
 0.00000000e+00]

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

Predicting on test set...
```



```
=== Final Evaluation: Bayes Optimal Classifier (Soft Voting) ===
Accuracy: 0.7090
Macro-averaged F1 score: 0.6146

Classification Report:
              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       0.80      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


Generating Confusion Matrix...
```



	BACKGROUND	CONCLUSIONS	METHODS	OBJECTIVE	RESULTS
BACKGROUND	1345	789	959	296	232
CONCLUSIONS	388	2575	644	61	903
METHODS	164	132	8789	66	746
OBJECTIVE	450	340	607	808	128
RESULTS	76	409	1372	8	7848

4. Discussion

Comparing my models, there was a clear improvement at each step.

My scratch model from **Part A** worked, but its F1 score wasn't as high as the `sklearn` models. This is probably because I used `CountVectorizer` (simple counts) and a default `alpha=1.0`.

The **Part B** `sklearn` model performed better after tuning. The `GridSearchCV` found the best `alpha` and `ngram_range`, and using `TfidfVectorizer` likely made a big difference.

The **Part C** BOC model was the most interesting. By combining five different types of models, it didn't have to rely on just one of them being perfect. Using soft voting with posterior weights let the ensemble trust the more confident models for a given prediction. This combined approach seems more robust than just relying on a single, highly-tuned algorithm.

---

5. Conclusion

This lab showed me how Naive Bayes is a strong baseline for text classification. I learned how to implement the core logic from scratch, including Laplace smoothing. I also saw how much easier and more powerful it is to use `sklearn`'s `Pipeline` and `GridSearchCV` tools. Finally, I saw how an ensemble (like the BOC) can combine diverse models to get a more robust and often more accurate result than any single model alone.