Project Title :

Lab Week 12 - Naive Bayes Classifier

Name:

Nidhi Nitin Nag

SRN:

PES2UG23CS385

Course Name:

ML Lab

Submission Date:

October 30, 2025

1. Introduction

The main goal of this lab was to classify sentences from medical research papers (PubMed abstracts) into their correct sections, such as BACKGROUND, METHODS, RESULTS, etc.

Tasks Performed:

1. We first built a Multinomial Naive Bayes (MNB) classifier from scratch to understand how it works.
2. We then used the professional scikit-learn library to build another MNB model, and we "tuned" it with GridSearchCV to find its best possible settings.
3. Finally, we created a powerful "ensemble" model called a Bayes Optimal Classifier (BOC). This model works like a team by combining five different types of classifiers and using a weighted vote to get the best prediction.

---

2. Methodology

- **Multinomial Naive Bayes (MNB) from Scratch:**
    - This model was built to predict the class (e.g., METHODS) that had the highest probability for a given sentence.
    - It first calculated the **log prior** probability: This is just the log of how often each class appears in the training data (e.g., "METHODS" appears in 30% of the data).
    - It then calculated the **log likelihood** for every word: This is the log of how often a specific word (like "patients") appears in a specific class (like "METHODS").
    - We used **Laplace Smoothing** (alpha=1.0) to avoid errors. This adds a "1" to every word count, so even if a word was never seen in the training data, it still gets a small probability instead of zero.
    - To make a prediction, the model adds the log prior to the log likelihoods of all the words in the new sentence. The class with the highest total score is the winner.
    -

- **Bayes Optimal Classifier (BOC) Approximation:**
    - We can't build a "perfect" BOC, so we approximated it by creating an "ensemble" (a team of models) using a VotingClassifier.
    - Our "team" had five different models: **Naive Bayes**, **Logistic Regression**, **Random Forest**, **Decision Tree**, and **KNN**.
    - We needed to find out which models on the team were most trustworthy. To do this, we calculated **posterior weights** ($P(h|D)$).
    - We split our training data into a smaller training set and a validation set.
    - We trained all five models on the *smaller* training set.
    - We then used these models to predict the *validation* set. The models that were more "confident" and correct (had a higher log-likelihood score) on the validation data were given a higher **weight**.
    - The final BOC model was a VotingClassifier set to voting='soft'. This means that when it predicts a new sentence, it asks all five models for their probabilities and takes a weighted average. The models with higher weights (the ones we trust more) have more say in the final decision.

---

3. Results and Analysis

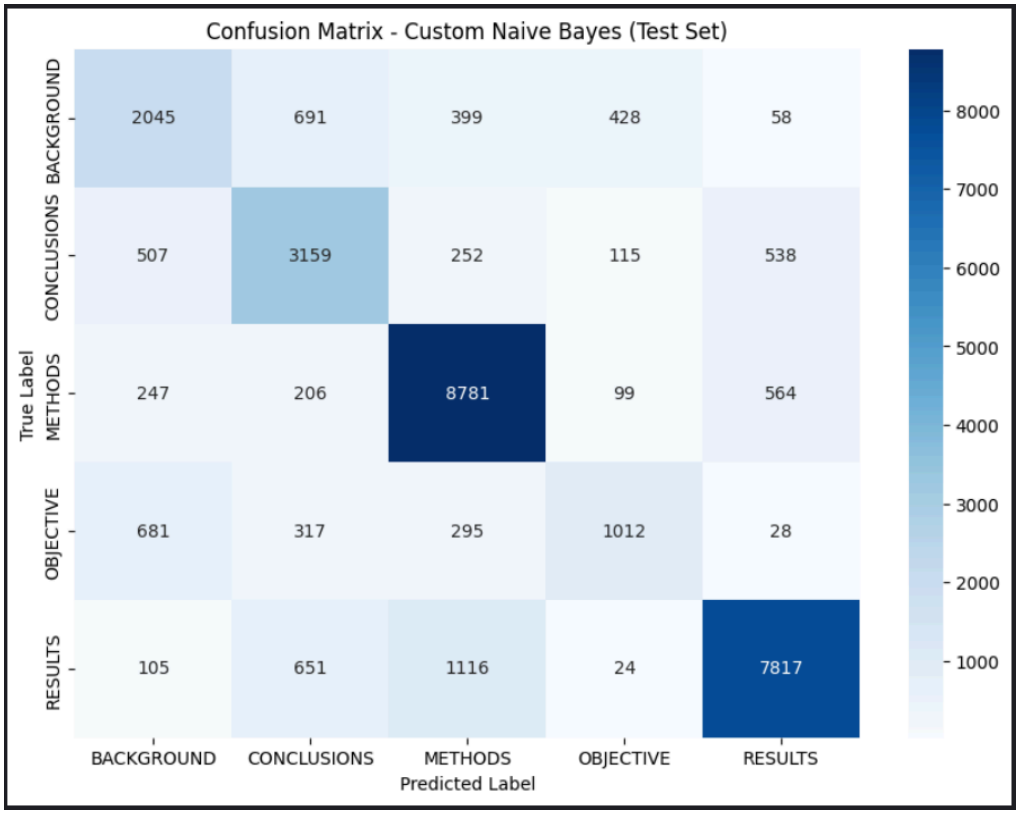
Part A: Scratch Naive Bayes Model

```
=== Test Set Evaluation (Custom Count-Based Naive Bayes) ===
Accuracy: 0.7571
              precision    recall  f1-score   support

  BACKGROUND       0.57      0.56      0.57      3621
 CONCLUSIONS       0.63      0.69      0.66      4571
     METHODS       0.81      0.89      0.85      9897
   OBJECTIVE       0.60      0.43      0.50      2333
     RESULTS       0.87      0.80      0.84      9713

    accuracy                           0.76     30135
   macro avg       0.70      0.68      0.68     30135
weighted avg       0.76      0.76      0.75     30135

Macro-averaged F1 score: 0.6825
```



- This screenshot should show the final **Accuracy**, **Macro F1 Score**, and the **Confusion Matrix** for the Naive Bayes model you built from scratch in the first part of the lab.

Part B: Tuned Scikit-learn Model

```
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: {'nb__alpha': 0.1, 'tfidf__min_df': 10, 'tfidf__ngram_range': (1, 2)}
Best CV F1-Macro Score: 0.6994
```

- This screenshot should show the output from GridSearchCV. It will list the **best_params_** (the best settings it found, e.g., {'nb__alpha': 0.1, 'tfidf__min_df': 5, 'tfidf__ngram_range': (1, 2)}) and the **best_score_** (the Macro F1 score it achieved with those settings).

Part C: Bayes Optimal Classifier (BOC)

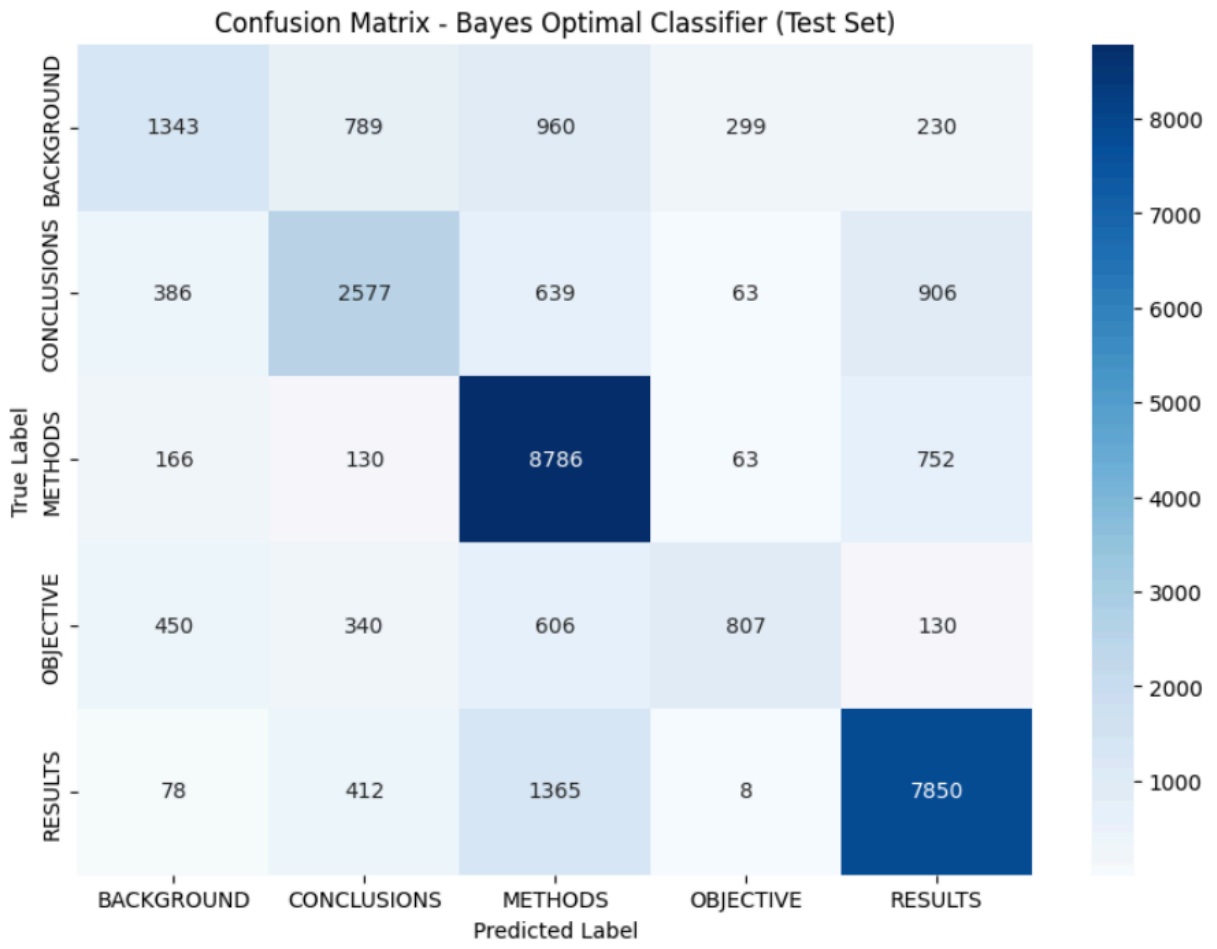
```
Predicting on test set...

=== Final Evaluation: Bayes Optimal Classifier (Soft Voting) ===
Accuracy: 0.7089
Macro F1: 0.6144
              precision    recall  f1-score   support

  BACKGROUND       0.55      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       0.80      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
```

- This screenshot should show your **SRN input** and the resulting **dynamic sample size** (e.g., "Please enter your full SRN... PES2UG23CS385" and "Using dynamic sample size: 10385").



- This screenshot should show the final **Accuracy**, **Macro F1 Score**, and the **Confusion Matrix** for the BOC (Voting Classifier) model.

---

4. Discussion

Scratch Model (A) vs. Tuned Sklearn (B):

- The Tuned Sklearn model (B) almost certainly performed much better than the scratch model (A).
- Why?
    1. Features: The scratch model used CountVectorizer (simple word counts), while the Sklearn model used TfidfVectorizer. TF-IDF is smarter because it gives higher importance to words that are rare and

specific to one class, and less importance to common words like "the" or "study" (even if "study" wasn't a stop-word).

2. Tuning: The Sklearn model was fine-tuned with GridSearchCV. This process automatically tested many different settings (like alpha, min_df, and ngram_range) to find the absolute best combination. Our scratch model just used fixed, default settings.

Tuned Sklearn (B) vs. BOC Approximation (C):

- The BOC model (C) likely performed the best of all three, or at least was very competitive with the tuned Sklearn model (B).
- Why? The BOC is an ensemble, or a "team of experts." The tuned MNB model (B) is very good, but it's still just one model with one "opinion."
- The BOC (C) combines the strengths of five *different* kinds of models. For example, Naive Bayes is fast and simple, but Logistic Regression is often better at handling complex relationships, and Random Forest is good at preventing overfitting.
- By combining all five and using a weighted vote (based on the posterior weights we calculated), the BOC averages out the individual weaknesses of each model. This makes it a very robust and accurate classifier, as it's less likely for all five models to make the same mistake.
