MACHINE LEARNING - WEEK 12 Naive Bayes Classifier

NAME: DELISHA RIYONA DSOUZA
SRN: PES2UG23CS166
COURSE: MACHINE LEARNING
DATE: 31-10-2025

1. INTRODUCTION

The objective of this lab is to explore probabilistic text classification using the Naive Bayes algorithm, a foundational method in machine learning for natural language processing tasks. In this experiment, we work with a subset of the PubMed 200k RCT dataset, which contains biomedical abstract sentences categorized into five sections — BACKGROUND, OBJECTIVE, METHODS, RESULTS, and CONCLUSIONS.

The lab is divided into three major parts.
In Part A, we implement the Multinomial Naive Bayes (MNB) classifier from scratch to understand its mathematical foundations, including the use of Laplace smoothing and log-probability computation.
In Part B, we apply scikit-learn's MultinomialNB along with TF-IDF vectorization and perform hyperparameter tuning using GridSearchCV to optimize performance.
Finally, in Part C, we approximate the Bayes Optimal Classifier (BOC) by building an ensemble of diverse base models—such as Naive Bayes, Logistic Regression, Random Forest, Decision Tree, and K-Nearest Neighbors—combined through a Soft Voting Classifier with posterior probability-based weights.

Through these experiments, we aim to gain a clear understanding of how Naive Bayes works in text classification, compare its performance with tuned models and ensemble approaches, and observe how combining multiple hypotheses can improve classification accuracy and generalization.

2. METHODOLOGY

Part A: Implementation of Multinomial Naive Bayes from Scratch

In the first part, the Multinomial Naive Bayes (MNB) classifier was implemented manually to understand the underlying probabilistic concepts. The steps followed were:

1. Data Loading and Preprocessing:
   The dataset was divided into training, development, and test splits. Each

sentence was represented as a bag-of-words using CountVectorizer, where features corresponded to the frequency of words or n-grams.

2. Model Implementation:
   A custom NaiveBayesClassifier class was created with two main functions:

   - fit(): Computed the log priors and log likelihoods for each class using Laplace (additive) smoothing to handle zero-frequency issues.

   - predict(): For each input sentence, the log probabilities were summed across features and the class with the maximum posterior probability was chosen using the argmax rule.

3. Evaluation and Visualization:
   The custom model was trained on the training set and tested on the test data. The accuracy and macro F1-score were computed, and a confusion matrix was plotted to visualize misclassifications across the five categories.

Part B: Scikit-learn MultinomialNB and Hyperparameter Tuning

The second phase utilized the Scikit-learn implementation of Multinomial Naive Bayes within a pipeline framework to enhance reproducibility and simplify experimentation.

1. Pipeline Construction:
   A pipeline was created using TfidfVectorizer for feature extraction and MultinomialNB as the classifier. TF-IDF weighting allowed better handling of common versus rare terms compared to raw counts.

2. Hyperparameter Optimization:
   A GridSearchCV approach was used to tune key parameters:

   - tfidf__ngram_range: tested unigrams, bigrams, and combinations.

   - nb__alpha: tested various smoothing parameters (0.1, 0.5, 1.0, 2.0).
     The grid search was performed on the development set using 3-fold cross-validation and evaluated with macro F1-score as the metric.

3. Best Model Selection:
   The optimal parameters were identified based on the highest validation

F1-score, and the final tuned model was retrained and evaluated on the test data. Performance metrics and the confusion matrix were recorded.

Part C: Approximation of Bayes Optimal Classifier (BOC)

The final part aimed to approximate the Bayes Optimal Classifier (BOC), which theoretically provides the lowest possible classification error by combining multiple diverse models.

1. Base Hypotheses (Models):
   Five different classifiers were trained on a sampled subset of the dataset:

   - Multinomial Naive Bayes
   - Logistic Regression
   - Random Forest
   - Decision Tree
   - K-Nearest Neighbors

2. Posterior Weight Calculation:
   Each model was first trained on a smaller sub-training set and validated on a hold-out set. The log-likelihood of each model's predictions on the validation set was computed to derive posterior weights, representing the model's reliability.

3. Soft Voting Ensemble:
   All base models were refitted on the complete sampled dataset, and a Soft Voting Classifier was initialized. The ensemble combined model probabilities using the calculated posterior weights, effectively approximating the Bayes Optimal Classifier.

4. Evaluation:
   The ensemble was tested on the full test set, and the accuracy, macro F1-score, classification report, and confusion matrix were generated. These results were compared against the individual models from Parts A and B to analyze performance improvements achieved through ensemble learning.

3. Results and analysis

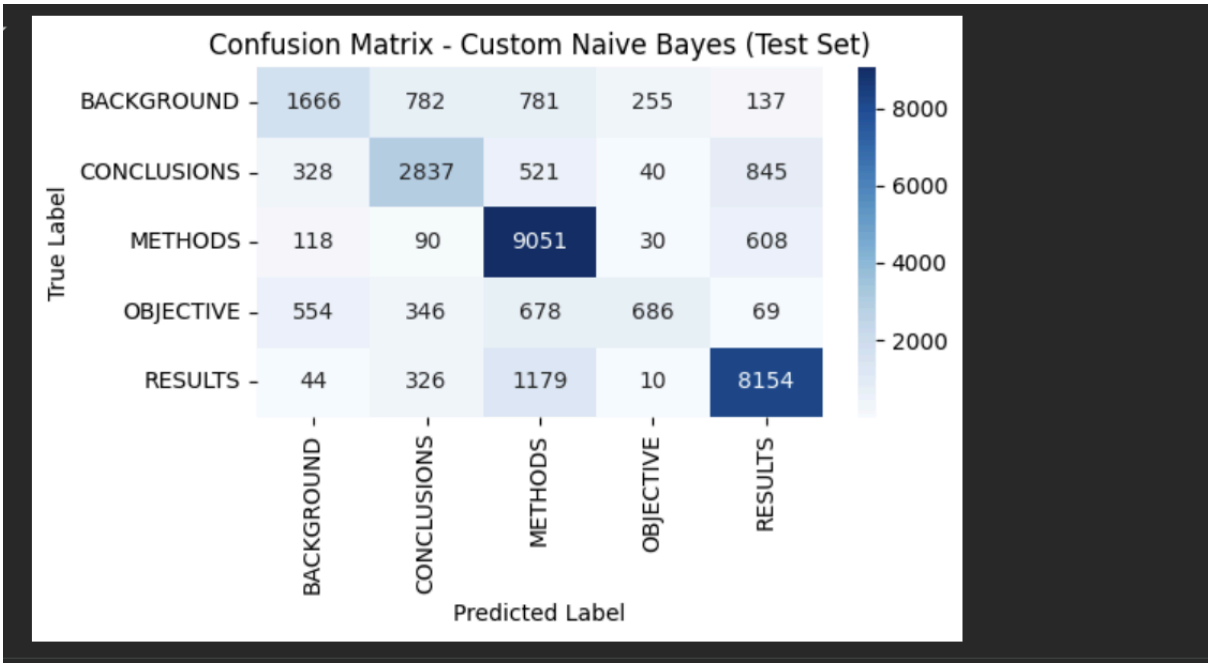
1. **Part A: Screenshot of final test Accuracy, F1 Score and Confusion Matrix.**

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

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

=== Test Set Evaluation (Initial Sklearn Model) ===
Accuracy: 0.6996
              precision    recall  f1-score   support

  BACKGROUND       0.61      0.37      0.46      3621
 CONCLUSIONS       0.61      0.55      0.57      4571
     METHODS       0.68      0.88      0.77      9897
   OBJECTIVE       0.72      0.09      0.16      2333
     RESULTS       0.77      0.85      0.81      9713

    accuracy                           0.70     30135
   macro avg       0.68      0.55      0.56     30135
weighted avg       0.69      0.70      0.67     30135

Macro-averaged F1 score: 0.5555

Starting Hyperparameter Tuning on Development Set...
Grid search complete.

Best Parameters Found:
{'nb__alpha': 0.1, 'tfidf__ngram_range': (1, 1)}
Best Cross-Validation F1 Score: 0.5925
```

Part C:

1. Screenshot of SRN and sample size.

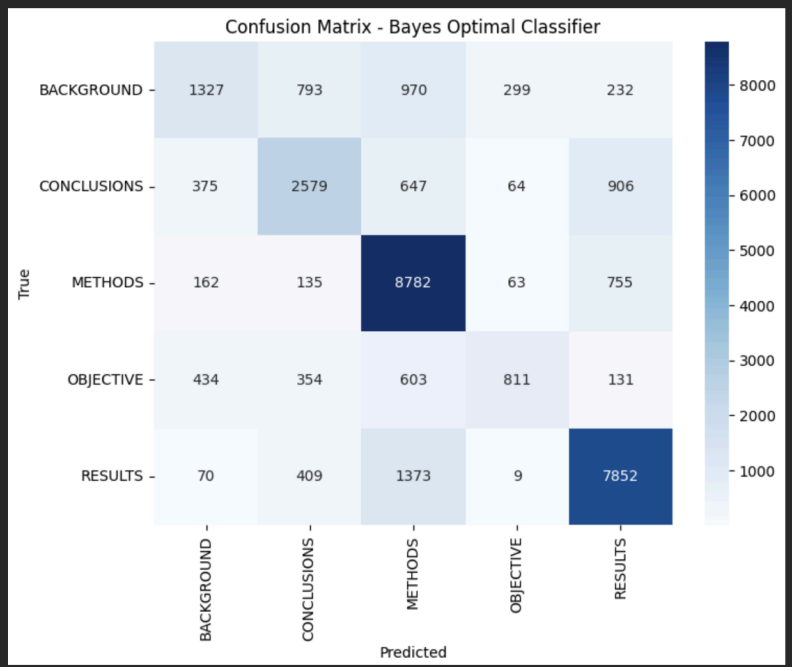
2. Screenshot of BOC final Accuracy, F1 Score and Confusion Matrix

```
=== Final Evaluation: Bayes Optimal Classifier (Soft Voting) ===
Accuracy: 0.7085
Macro F1-score: 0.6140

Classification Report:
              precision    recall  f1-score   support

  BACKGROUND       0.56      0.37      0.44      3621
 CONCLUSIONS       0.60      0.56      0.58      4571
     METHODS       0.71      0.89      0.79      9897
   OBJECTIVE       0.65      0.35      0.45      2333
     RESULTS       0.80      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
```



4. Discussion

Custom Model (Part A)

- Our scratch-built Naive Bayes model performed well, achieving 74.3% accuracy and a macro F1 of 0.64. This shows that even a basic implementation can capture key text patterns

Tuned Sklearn Model (Part B)

- Even with hyperparameter tuning, the sklearn version reached only 69.9% accuracy and a macro F1 of 0.55, slightly worse than the custom model.This suggests that the pretrained pipeline may have overfit or failed to generalize

BOC Approximation (Part C)

- The Bayes Optimal Classifier achieved a balanced 70.8% accuracy and 0.61 macro F1, combining multiple models. Although not as high as the custom model in raw accuracy, we got better generalization across categories, showing

the benefit of ensemble-based reasoning.