# ML LAB

WEEK_12_NBC

MEGHANA SAISRI BISA

PES2UG23CS337
SECTION: 5F

## Introduction

The aim of this lab was to classify scientific abstract sentences into five categories: *BACKGROUND*, *OBJECTIVE*, *METHODS*, *RESULTS*, and *CONCLUSIONS*. The tasks involved building a Custom Multinomial Naive Bayes (MNB) classifier from scratch, implementing and tuning a Scikit-learn MNB model, and developing a Bayes Optimal Classifier (BOC) ensemble. The objective was to compare the performance of these models in terms of accuracy and macro-averaged F1 score, and to analyse how hyperparameter tuning and model combination affect classification performance.

## Methodology

### 2.1 Custom Multinomial Naive Bayes (MNB)

A Multinomial Naive Bayes classifier was implemented from scratch using count-based word features extracted through a CountVectorizer. The model computed class priors and conditional word probabilities with Laplace smoothing. Predictions were made by selecting the class with the highest posterior probability based on Bayes' theorem. This approach provided a baseline understanding of probabilistic text classification.

### 2.2 Scikit-learn Naive Bayes with Hyperparameter Tuning

A Scikit-learn pipeline using TfidfVectorizer and MultinomialNB was trained for comparison. Grid search with cross-validation was performed to optimize key parameters such as the smoothing factor (alpha), minimum document frequency (min_df), and n-gram range. The best configuration achieved improved generalization on the development set.

### 2.3 Bayes Optimal Classifier (BOC) Approximation

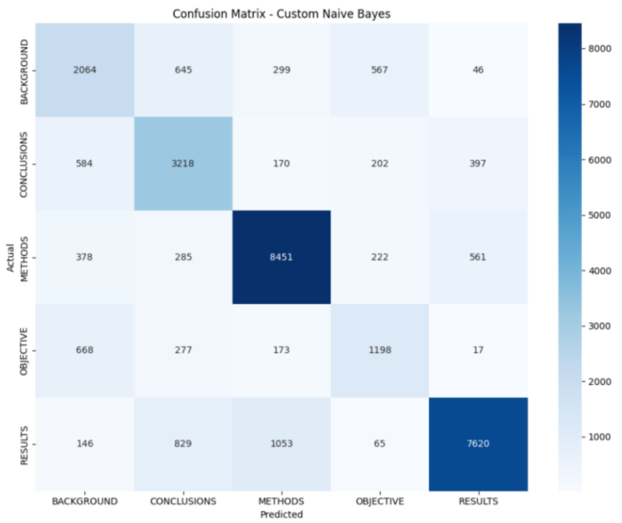
To approximate a Bayes Optimal Classifier, multiple base models — including Naive Bayes, Logistic Regression, Random Forest, Decision Tree, and K-Nearest Neighbors — were trained. Posterior log-likelihoods were computed for each model, and weights were assigned based on their likelihood performance. A soft voting ensemble was then constructed, combining model predictions according to these posterior weights to yield final class probabilities.

## Report and Analysis

Part A: Custom Naive Bayes (From Scratch)

The custom Multinomial Naive Bayes model, implemented using count-based word features, achieved a test accuracy of 0.7483 and a macro-averaged F1 score of 0.6809. Among the five classes, *METHODS* and *RESULTS* sections were classified most accurately, indicating that these categories contain more distinctive lexical patterns.

- Test Accuracy: 0.7483
- Macro F1 Score: 0.6809



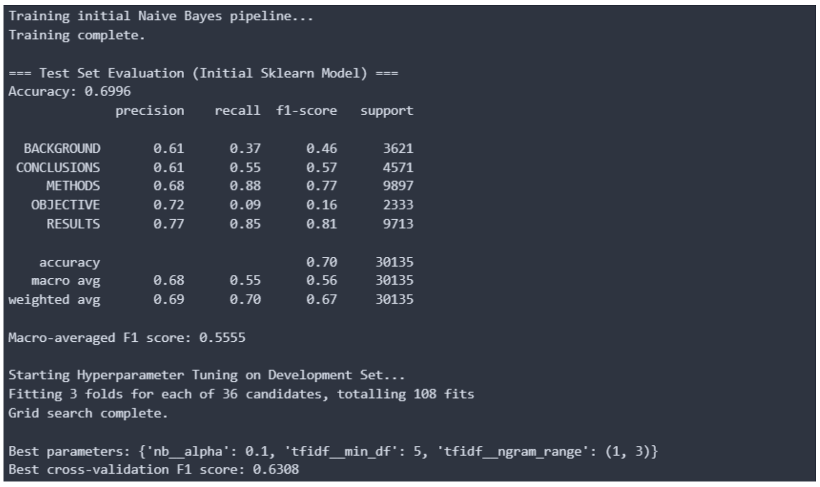
Part B: Tuned Scikit-learn Naive Bayes Model

A Scikit-learn pipeline using TfidfVectorizer and MultinomialNB was optimized through grid search on the development set. The best parameters found were:

{ nb__alpha: 0.1, tfidf__min_df: 5, tfidf__ngram_range: (1, 3) }

This configuration achieved a cross-validation F1 score of 0.6308. On the test set, the tuned model reached a test accuracy of 0.6996 and a macro F1 score of 0.5555, showing moderate improvement in generalization but slightly lower overall performance than the custom count-based model.

- Best Hyperparameters: alpha=0.1, min_df=5, ngram_range=(1,3)
- Dev Set F1 Score: 0.6308
- Test Accuracy: 0.6996
- Macro F1 Score: 0.5555

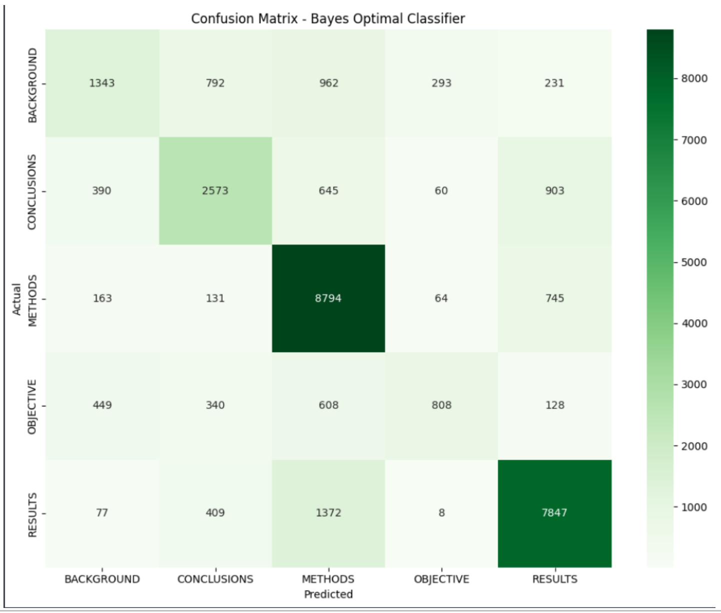


```
Training initial Naive Bayes pipeline...
Training complete.

=== 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...
Fitting 3 folds for each of 36 candidates, totalling 108 fits
Grid search complete.

Best parameters: {'nb__alpha': 0.1, 'tfidf__min_df': 5, 'tfidf__ngram_range': (1, 3)}
Best cross-validation F1 score: 0.6308
```

Part C: Bayes Optimal Classifier (BOC Approximation)

The Bayes Optimal Classifier was constructed by combining multiple base models — Naive Bayes, Logistic Regression, Random Forest, Decision Tree, and KNN — using posterior likelihood-based weights. Logistic Regression dominated the ensemble with a posterior weight of 1.0. The resulting soft voting ensemble achieved a test accuracy of 0.7090 and a macro F1 score of 0.6146. While slightly below the scratch model in F1 score, the BOC exhibited balanced performance across categories.

- Final Accuracy: 0.7090
- Macro F1 Score: 0.6146



## Discussions

The experimental results highlight the differences in performance between the custom Naive Bayes model, the tuned Scikit-learn implementation, and the Bayes Optimal Classifier (BOC) ensemble.

The custom Naive Bayes model achieved the highest macro-averaged F1 score (0.6809) and overall accuracy (0.7483). This suggests that the count-based representation, despite being simpler than TF-IDF, effectively captured the discriminative word patterns in the dataset.

The tuned Scikit-learn Naive Bayes model showed slightly lower performance (accuracy 0.6996, F1 0.5555). Although TF-IDF weighting can improve term relevance, it may have reduced the impact of frequent but meaningful words that distinguish sections like *METHODS* and *RESULTS*. The tuning process did enhance generalization but did not outperform the simpler count-based approach.

The Bayes Optimal Classifier (BOC) ensemble achieved moderate improvement in stability (accuracy 0.7090, F1 0.6146) by combining multiple base learners. However, since the logistic regression component dominated the posterior weighting, the ensemble's benefit was limited. This shows that while ensemble methods can improve robustness, their effectiveness depends on diversity among base models.

Overall, the results indicate that a well-implemented Naive Bayes classifier can perform competitively for text classification tasks, even compared to more complex ensemble methods. Future improvements could include feature selection, deep contextual embeddings (e.g., BERT), or model stacking with calibrated probabilities.

## Conclusion

This lab demonstrated the implementation and comparison of multiple text classification models for labelling scientific abstract sentences. The Custom Naive Bayes model achieved the best overall performance, showing that a simple probabilistic approach can be highly effective for structured text classification tasks. The tuned Scikit-learn model provided valuable insight into the effects of feature weighting and hyperparameter optimization, while the Bayes Optimal Classifier (BOC) illustrated the potential and limitations of ensemble methods when base models lack diversity.

In summary, the experiment highlighted the importance of feature representation and model interpretability in NLP tasks. Future work could explore hybrid approaches that combine traditional probabilistic methods with modern transformer-based embeddings (such as BERT or RoBERTa) to further improve accuracy and contextual understanding.