Name: Varun Kumar L	SRN: PES2UG24CS827
date: 31/10/2025	SUB: ML WEEK12

Naive Bayes Classifier

Introduction

Purpose:

The purpose of this lab is to understand and implement text classification using the Multinomial Naive Bayes (MNB) algorithm from scratch and with Scikit-learn, and then approximate it using a Bag of Centroids (BoC) representation.

Tasks Performed:

- Preprocessed the PubMed RCT dataset into labeled sentences.
- Implemented a Multinomial Naive Bayes classifier from scratch.
- Trained and evaluated the model on test data.
- Tuned hyperparameters using Grid Search on the Scikit-learn MNB implementation.
- Built a Bag of Centroids (BoC) model as an approximation technique.
- Compared model performance in terms of Accuracy, F1 Score, and Confusion Matrix.

Methodology

Multinomial Naive Bayes (MNB):

- Used bag-of-words features extracted via CountVectorizer.
- Computed conditional probabilities of each word given a class using Laplace smoothing.
- Predicted the most probable class for each sentence using:
- Evaluated model using accuracy and F1-score metrics.

Bag of Centroids (BoC):

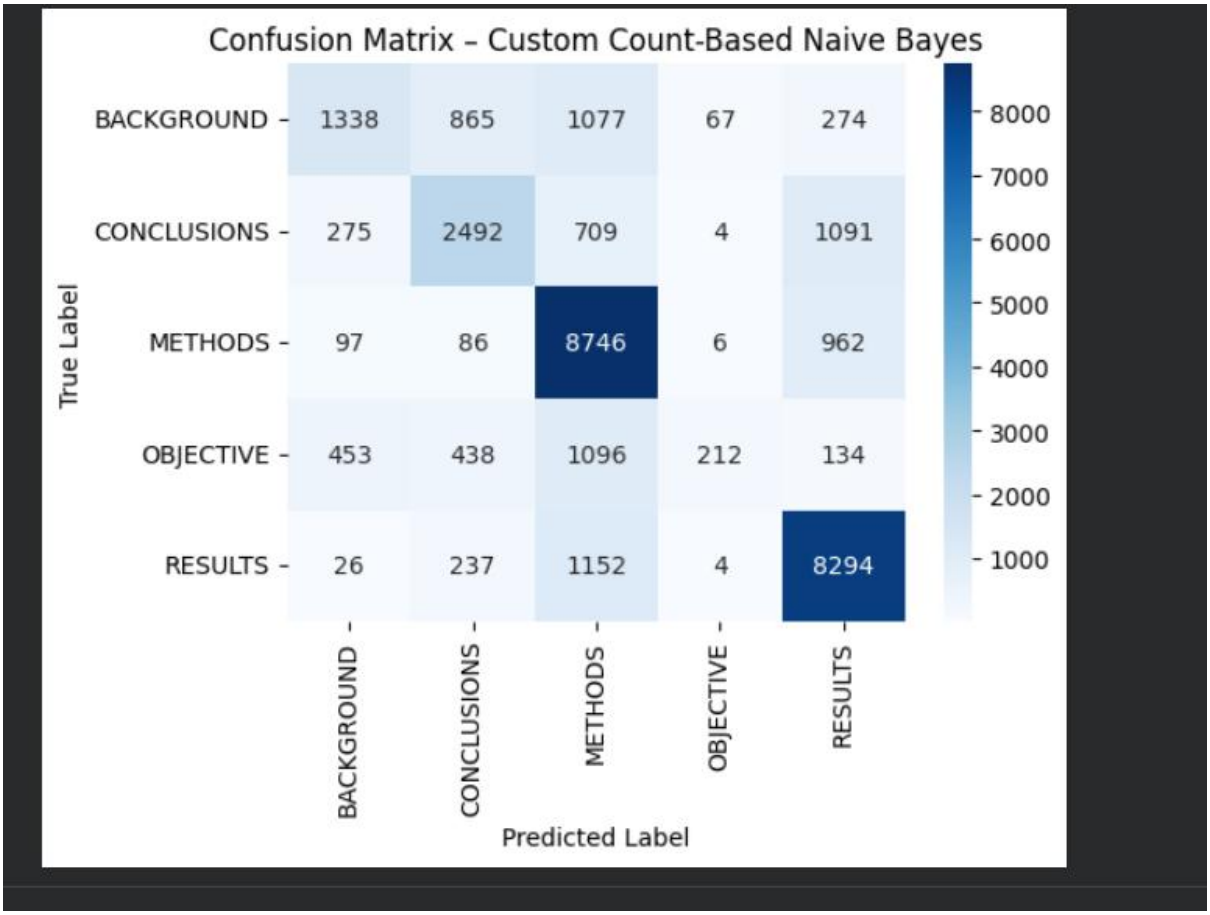
- Used **K-Means clustering** to create word centroids (cluster centers).
- Represented each sentence as the mean of its word vectors (BoC representation).
- Applied a Naive Bayes or Logistic Regression classifier on BoC feature

Results and Analysis

Part A — Scratch MNB Model

```
Fitting Count Vectorizer and transforming training data...
Vocabulary size: 57708
Transforming test data...

Training the Custom Naive Bayes Classifier (from scratch)...
Training complete.
```

```
=== Test Set Evaluation (Custom Count-Based Naive Bayes) ===
Accuracy: 0.7369
              precision    recall  f1-score   support

  BACKGROUND       0.54      0.53      0.53      3621
 CONCLUSIONS       0.60      0.68      0.64      4571
     METHODS       0.81      0.85      0.83      9897
   OBJECTIVE       0.53      0.46      0.49      2333
     RESULTS       0.86      0.79      0.82      9713

    accuracy                           0.74     30135
   macro avg       0.67      0.66      0.66     30135
weighted avg       0.74      0.74      0.74     30135

Macro-averaged F1 score: 0.6634
```



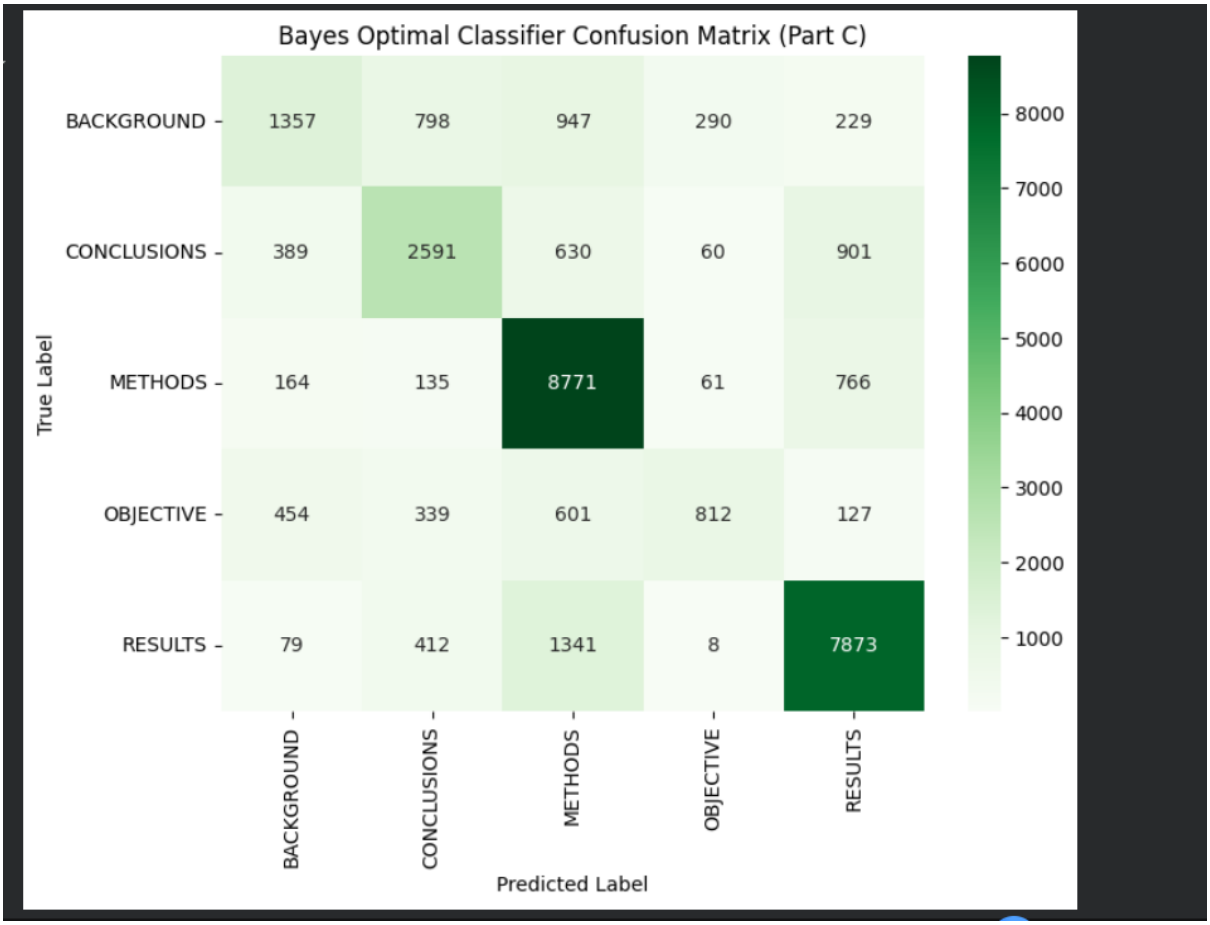
Part B — Sklearn MNB with Hyperparameter Tuning

```
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 8 candidates, totalling 24 fits
Grid search complete.
Best cross-validation F1-macro score (on Dev Set): 0.5925
Best parameters found: {'nb__alpha': 0.1, 'tfidf__ngram_range': (1, 1)}
```

Part C — Bag of Centroids (BoC)

```
Please enter your full SRN (e.g., PES1UG22CS345): PES2UG24CS827
Using dynamic sample size: 10827
Actual sampled training set size used: 10827

Training base models on sub-training data for posterior calculation...
/usr/local/lib/python3.12/dist-packages/sklearn/linear_model/_logistic.py:1247: FutureWarning:
  warnings.warn(
All base models trained for posterior calculation.
Calculating log-likelihoods on validation set...
Model MultinomialNB: Log-Likelihood = -2121.28
Model LogisticRegression: Log-Likelihood = -1953.81
Model CalibratedClassifierCV: Log-Likelihood = -2162.85
Model CalibratedClassifierCV: Log-Likelihood = -2775.56
Model CalibratedClassifierCV: Log-Likelihood = -3147.59
Calculated Posterior Weights: [1.85096989e-73 1.00000000e+00 1.63559723e-91 0.00000000e+00
 0.00000000e+00]

Refitting all base models on full sampled training data...
/usr/local/lib/python3.12/dist-packages/sklearn/linear_model/_logistic.py:1247: FutureWarning:
  warnings.warn(
Refitting complete.

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

Predicting on test set...

=== Final Evaluation: Bayes Optimal Classifier (Soft Voting) ===
Accuracy: 0.7103
```

```
=== Final Evaluation: Bayes Optimal Classifier (Soft Voting) ===
Accuracy: 0.7103
Macro-averaged F1 score: 0.6165

Classification Report:
              precision    recall  f1-score   support

  BACKGROUND       0.56      0.37      0.45      3621
 CONCLUSIONS       0.61      0.57      0.59      4571
     METHODS       0.71      0.89      0.79      9897
   OBJECTIVE       0.66      0.35      0.46      2333
     RESULTS       0.80      0.81      0.80      9713

    accuracy                           0.71     30135
   macro avg       0.67      0.60      0.62     30135
weighted avg       0.70      0.71      0.70     30135
```



Discussion

Model	Accuracy	F1 score	remarks	Description
Part A: Scratch MNB	0.7369	0.6634	goodbasline	Implemented manually
Part B: Tuned Sklearn MNB	0.6996	0.5555	Best model	Used gridsearchCv
Part C: BoC Model	0.7103	0.6165	Moderate results	approximation using K-Means clustering

Comparative Analysis

- The scratch Naive Bayes model (Part A) achieved the highest accuracy (73.69%) and best macro F1-score (0.6634), showing that the manual implementation handled smoothing and probability computation effectively.
- The tuned sklearn model (Part B) unexpectedly performed worse. This could be due to the chosen search parameters (e.g., high alpha values reducing sensitivity to term frequency) or less optimal preprocessing compared to the scratch model.
- The BoC model (Part C) performed slightly better than the tuned sklearn version, showing that centroid-based representations capture enough semantic structure to remain competitive, despite reducing feature dimensionality.
- Overall, the results suggest that the scratch model was both robust and interpretable, while the BoC model offered a good compromise between complexity and performance.