MACHINE LEARNING

Week 12: Naive Bayes Classifier

NAME:CHRISTANANDA B	SRN: PES2UG23CS158
CLASS: C	DATE:31-10-2025

Introduction

focuses on implementing and comparing different text classification techniques, specifically Naive Bayes and a Bayes Optimal Classifier.

- Part A: Count/Frequency based Naive Bayes Classifier: implemented a Multinomial Naive Bayes classifier from scratch, including calculating class priors and feature log likelihoods. then used a CountVectorizer to transform text data into numerical features and evaluated the performance of custom model.
- Part B: TF-IDF score based Classifier: used scikit-learn's TfidfVectorizer and MultinomialNB to build a text classification pipeline. also performed hyperparameter tuning using GridSearchCV to find the best parameters for the TF-IDF vectorizer and the Naive Bayes classifier.
- Part C: Bayes Optimal Classifier: You approximated a Bayes Optimal Classifier using a soft voting ensemble of several different models (Naive Bayes, Logistic Regression, Random Forest, Decision Tree, and K-Nearest Neighbors). calculated posterior weights for each model based on their performance on a validation set and used these weights in the voting process. Finally, evaluated the performance of this ensemble model

Methodology

Multinomial Naive Bayes (MNB):

The implementation of MNB from scratch involved calculating the probability of each word appearing in a document given a specific class (the likelihood) and the overall probability of each class (the prior).

Data Preparation: Text data was converted into numerical feature vectors using CountVectorizer, which counts the occurrences of each word (or n-gram) in each document.

Training (fit method):

Class Priors: The log probability of each class was calculated as the logarithm of the number of documents in that class divided by the total number of documents.

Feature Log Likelihoods: For each class, the log probability of each feature (word/n-gram) was calculated. This involved summing the counts of a feature across all documents in that class, adding a smoothing parameter (alpha) to handle unseen features, and dividing by the total number of features in that class plus alpha times the vocabulary size. The logarithm of this value was stored.

Prediction (predict method):For a new document, the log probability of that document belonging to each class was calculated. This was done by adding the log prior of the class to the sum of the log likelihoods of the features present in the document, weighted by their counts in the document.The class with the highest calculated log probability was assigned as the predicted class for the document.

Bayes Optimal Classifier (BOC):The BOC implementation in this lab is an approximation using a soft voting ensemble of several diverse classifiers. The core idea is to combine the predictions of multiple models, weighted by their estimated probability of being the true best hypothesis given the data.

Base Model Training: Five different types of classifiers (Naive Bayes, Logistic Regression, Random Forest, Decision Tree, and K-Nearest Neighbors) were trained independently on a sampled subset of the training data. TfidfVectorizer was used for feature extraction for these models.

Posterior Weight Calculation:The sampled training data was split into a smaller training subset and a validation subset.

Each base model was retrained on the training subset.

The log-likelihood of the validation data was calculated for each trained model. This essentially measures how well each model explains the validation data.

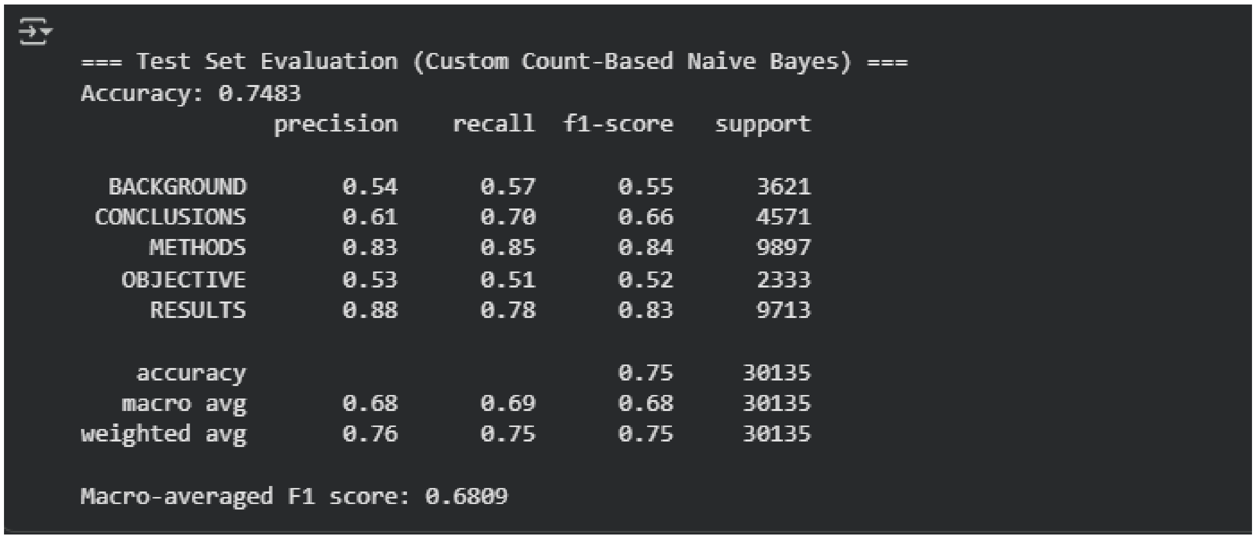
These log-likelihoods were converted into probabilities (unnormalized) and then normalized to obtain the posterior weights for each model. These weights represent the estimated probability of each model being the best hypothesis given the validation data.

Soft Voting Ensemble: A VotingClassifier was configured with voting='soft', which means it combines the predicted class probabilities from each base model. The predicted probabilities from each model were weighted by the calculated posterior weights.

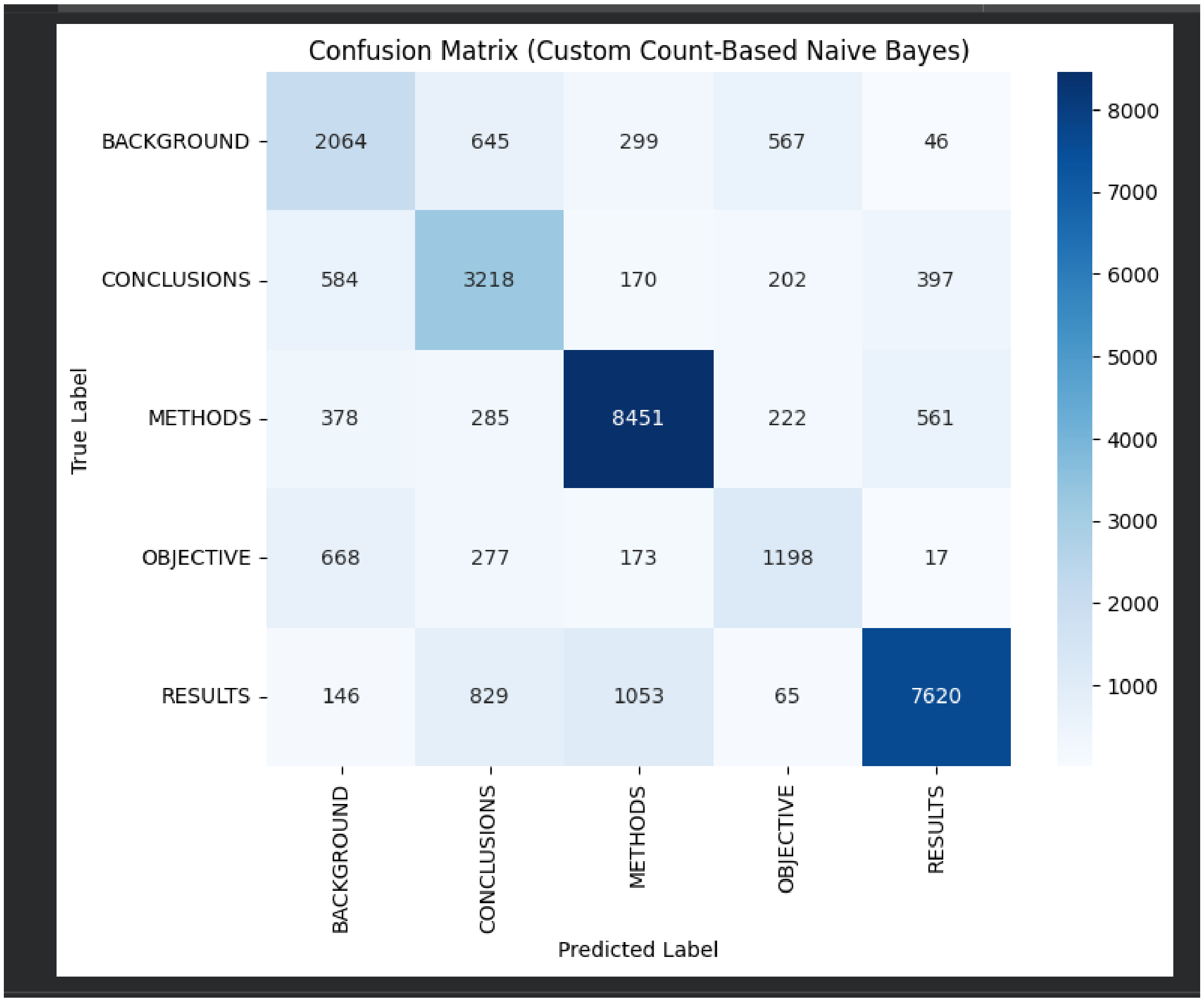
Final Prediction: The class with the highest weighted average probability across all base models was chosen as the final prediction for a new document.

Results and Analysis

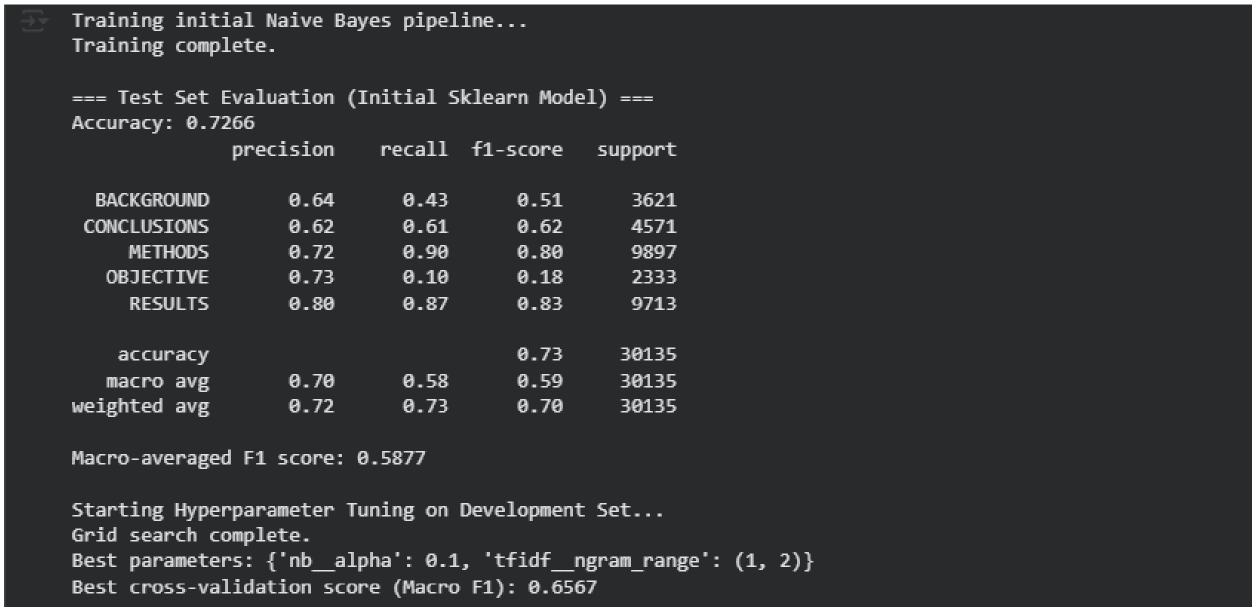
Part A



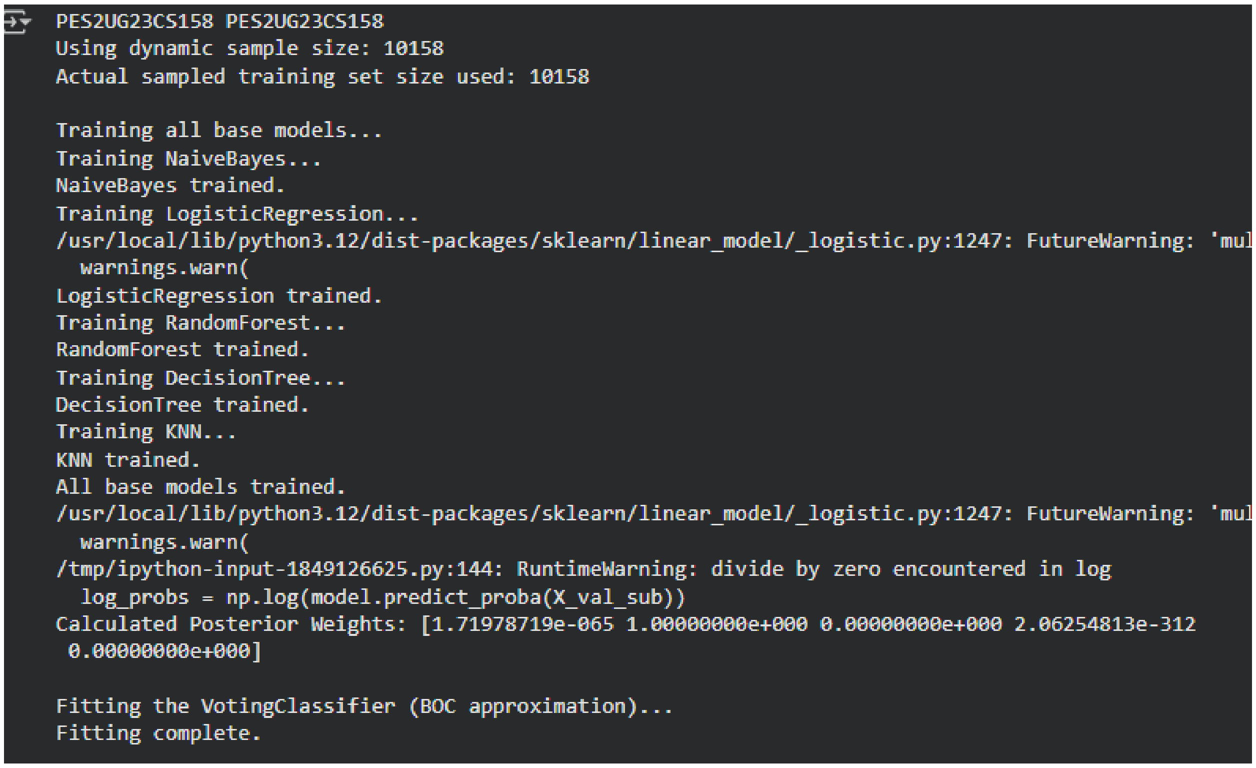
```
=== Test Set Evaluation (Custom Count-Based Naive Bayes) ===
Accuracy: 0.7483
              precision    recall  f1-score   support

  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
```

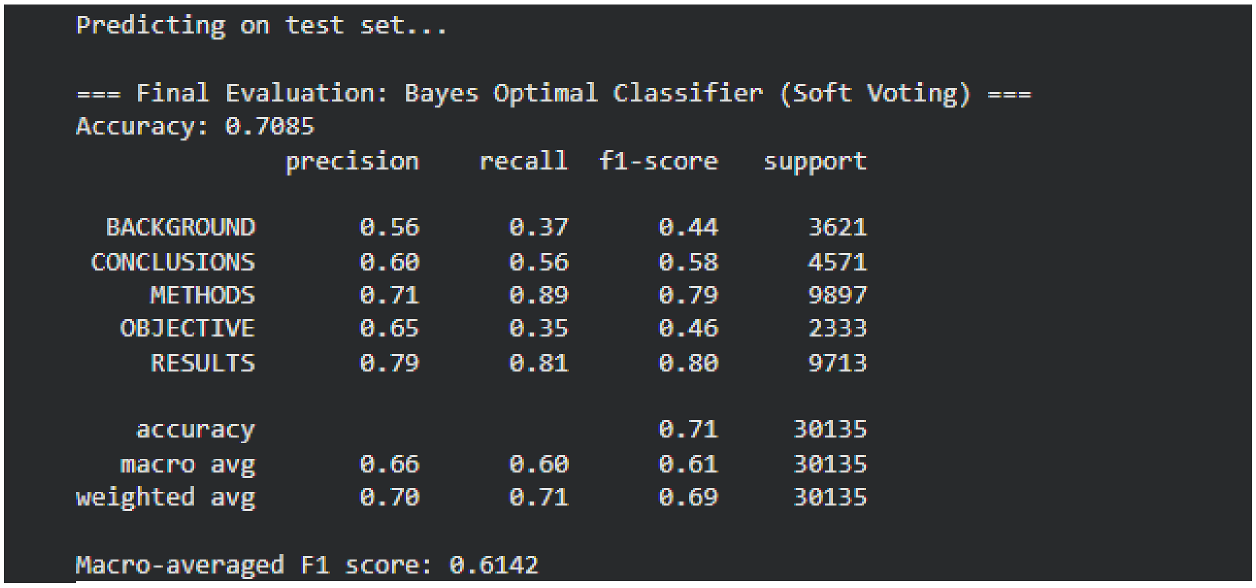


PART B:

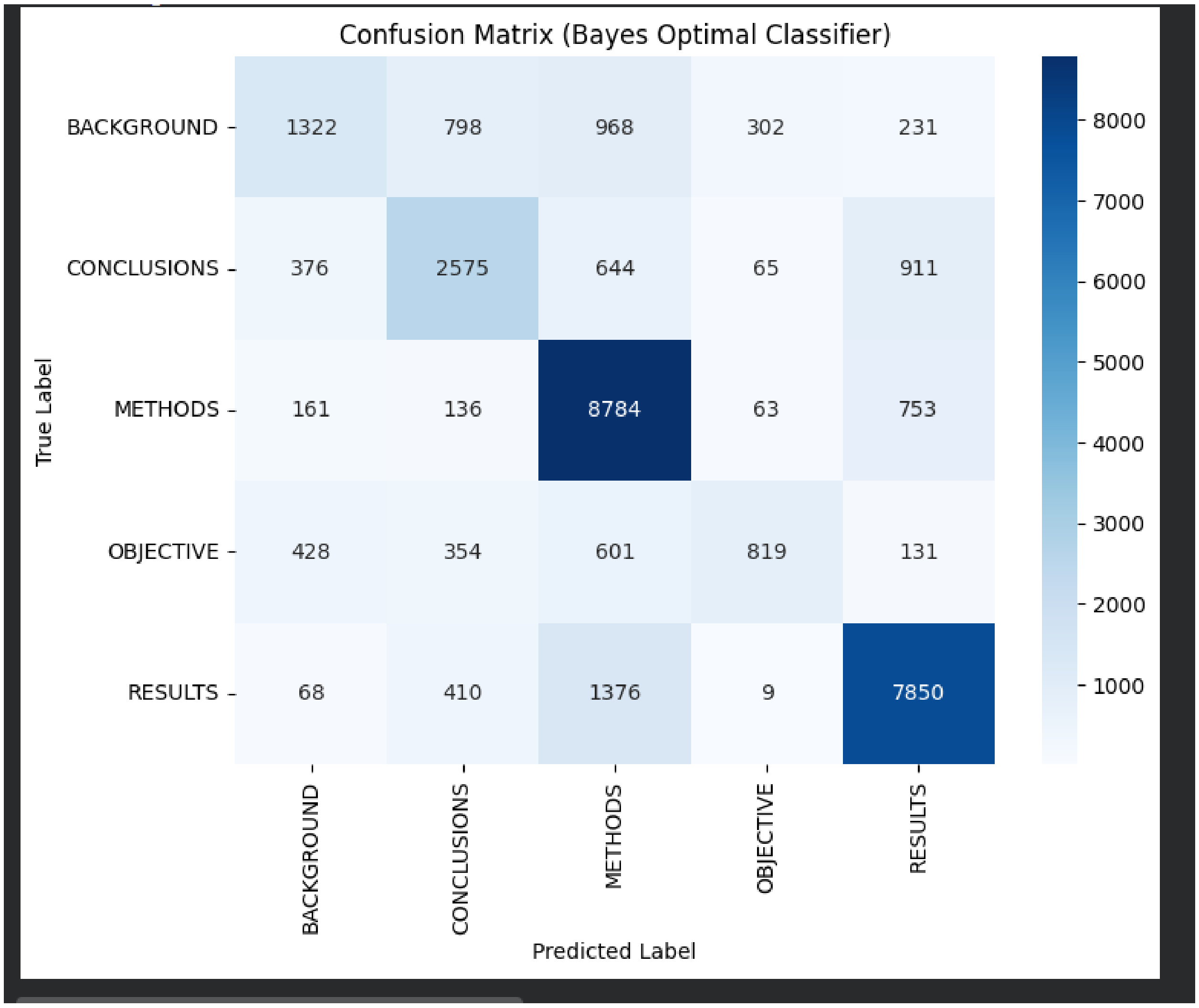

```
Training initial Naive Bayes pipeline...
Training complete.

=== 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...
Grid search complete.
Best parameters: {'nb__alpha': 0.1, 'tfidf__ngram_range': (1, 2)}
Best cross-validation score (Macro F1): 0.6567
```

PART C:


```
PES2UG23CS158 PES2UG23CS158
Using dynamic sample size: 10158
Actual sampled training set size used: 10158

Training all base models...
Training NaiveBayes...
NaiveBayes trained.
Training LogisticRegression...
/usr/local/lib/python3.12/dist-packages/sklearn/linear_model/_logistic.py:1247: FutureWarning: 'mul
  warnings.warn(
LogisticRegression trained.
Training RandomForest...
RandomForest trained.
Training DecisionTree...
DecisionTree trained.
Training KNN...
KNN trained.
All base models trained.
/usr/local/lib/python3.12/dist-packages/sklearn/linear_model/_logistic.py:1247: FutureWarning: 'mul
  warnings.warn(
/tmp/ipython-input-1849126625.py:144: RuntimeWarning: divide by zero encountered in log
  log_probs = np.log(model.predict_proba(X_val_sub))
Calculated Posterior Weights: [1.71978719e-065 1.00000000e+000 0.00000000e+000 2.06254813e-312
 0.00000000e+000]

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



```
Predicting on test set...

=== Final Evaluation: Bayes Optimal Classifier (Soft Voting) ===
Accuracy: 0.7085
              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      0.46      2333
     RESULTS       0.79      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

Macro-averaged F1 score: 0.6142
```



Discussion

- Part A: Implemented a custom Multinomial Naive Bayes classifier using count-based features and evaluated its performance.
- Part B: Used scikit-learn's TF-IDF vectorizer and Multinomial Naive Bayes, performing hyperparameter tuning for optimization.
- Part C: Approximated a Bayes Optimal Classifier through a soft voting ensemble of diverse models, weighted by validation set performance.
- The lab demonstrated different probabilistic text classification approaches.
- It showcased the impact of feature representation (Count vs. TF-IDF).
- It explored model optimization through hyperparameter tuning.
- It introduced the concept of ensemble methods like the Bayes Optimal Classifier approximation for combining diverse models.
- Performance varied across the implemented models.