Naive Bayes Classifier

Parth Upadhyay
Roll No: 25CS60R35
Technical Report

1. Objectives

- Understand the theoretical foundations of the Naive Bayes classifier.
- Implement Gaussian Naive Bayes from scratch in Python.
- Apply the algorithm to a spam detection dataset and tune its hyperparameters.
- Analyze the independence assumption by artificially duplicating predictive features.
- Evaluate the impact on model metrics when correlating features.

2. Methodology

The Naive Bayes classifier assumes that features are conditionally independent given the class. Here, a Gaussian variant is implemented, suitable for continuous variables. The training process estimates the mean and variance per class for each feature and computes class priors. Smoothing via an "alpha" hyperparameter prevents divide-by-zero errors and helps generalization.

Algorithm:

1. Estimate class priors $P(y)$ and per-feature means and variances for each class from training data.
2. For prediction, compute the likelihood under the Gaussian assumption for each feature and sum log probabilities.
3. Combine likelihoods to obtain overall class log-probabilities and select the highest.

3. Implementation

Code Snippet (Gaussian Naive Bayes):

```
class NaiveBayesClassifier:
    def __init__(self, alpha=1):
```

```
        self.alpha = alpha
        ...
    def fit(self, X, y):
        ...
    def predict(self, X):
        ...
```

Hyperparameter tuning is performed over several "alpha" values, and performance metrics (accuracy, precision, recall, F1) are calculated. Confusion matrices for both train and test sets are presented.

4. Results

4.1. Hyperparameter Tuning

The smoothing parameter α notably affects the variance of the Gaussian estimates. Low values (e.g., 0.001) provide optimal performance:

- Best test accuracy: **81.43%** for $\alpha = 0.001$
- Regularization reduces overfitting; too much smoothing reduces discriminative power.
- Confusion matrix (test): TP=374, FP=155, FN=16, TN=376

4.2. Duplicate (Correlated) Features

To investigate independence, copies of a strongly predictive feature are artificially added:

- Adding duplicates increases both training and test accuracy modestly (**+1.2%** after four copies). The likelihood term is mathematically raised to additional powers, making Naive Bayes overconfident in this feature.
- Overconfidence stems from violating the independence assumption:

$$\log P(y) + n \log P(x_k|y) + \sum_{i \neq k} \log P(x_i|y)$$

4.3. Metrics

α	Train Acc.	Test Acc.	Precision	Recall	F1
0.0001	80.00%	80.02%	84.94%	80.00%	82.39%
0.001	80.65%	81.43%	85.47%	81.43%	83.29%
0.01	79.62%	79.70%	85.12%	79.70%	81.86%
0.1	76.88%	76.66%	84.12%	76.66%	79.91%
1	80.30%	77.74%	81.10%	77.74%	80.70%
10	68.02%	63.74%	72.61%	63.74%	66.16%

5. Discussion

Hyperparameter Smoothing

Very small α (e.g., 0.0001) barely smooths the variances, resulting in overfitting. Moderate smoothing ($\alpha \approx 0.001$) stabilizes estimates and improves generalization. Large α makes features less informative and harms both recall and precision.

Effects of Hyperparameters

Q: Which value of α is most suitable for your dataset and why?
A: $\alpha = 0.001$ gives the highest accuracy and balanced metrics.

Q: How does smoothing influence overfitting?
A: Very small α barely smooths variance and can overfit; moderate α stabilizes Gaussian estimates and improves generalization; large α inflates variance, reducing informativeness.

Effects of Duplicates

Q: How do duplicate features affect performance?
A: Duplicating a predictive feature reinforces its effect, slightly increasing accuracy but also confidence, since Naive Bayes double-counts identical information.

Mathematically:

$$\log P(y) + [\log P(x_k|y) + \log P(x'_k|y)] + \sum_{i \neq k} \log P(x_i|y)$$

This doubles the contribution of x_k:

$$\log P(y) + 2 \log P(x_k|y) + \sum_{i \neq k} \log P(x_i|y)$$

So the model becomes overconfident.

Naive Bayes Visualizations

This section presents figures generated from the `naiveBayes_Tutorial.ipynb` notebook, including data distributions, decision boundaries, and learning curves.

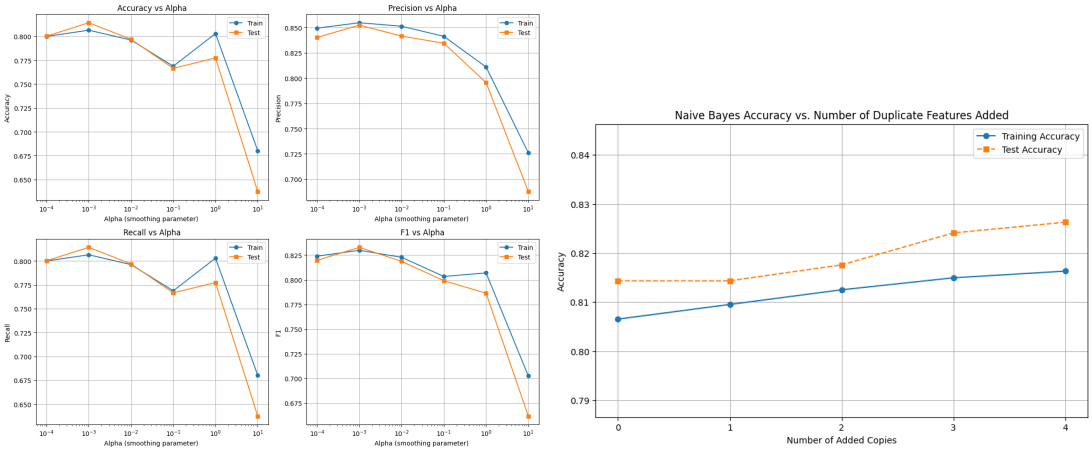

Figure 1: Final Naive Bayes results and summary plots.

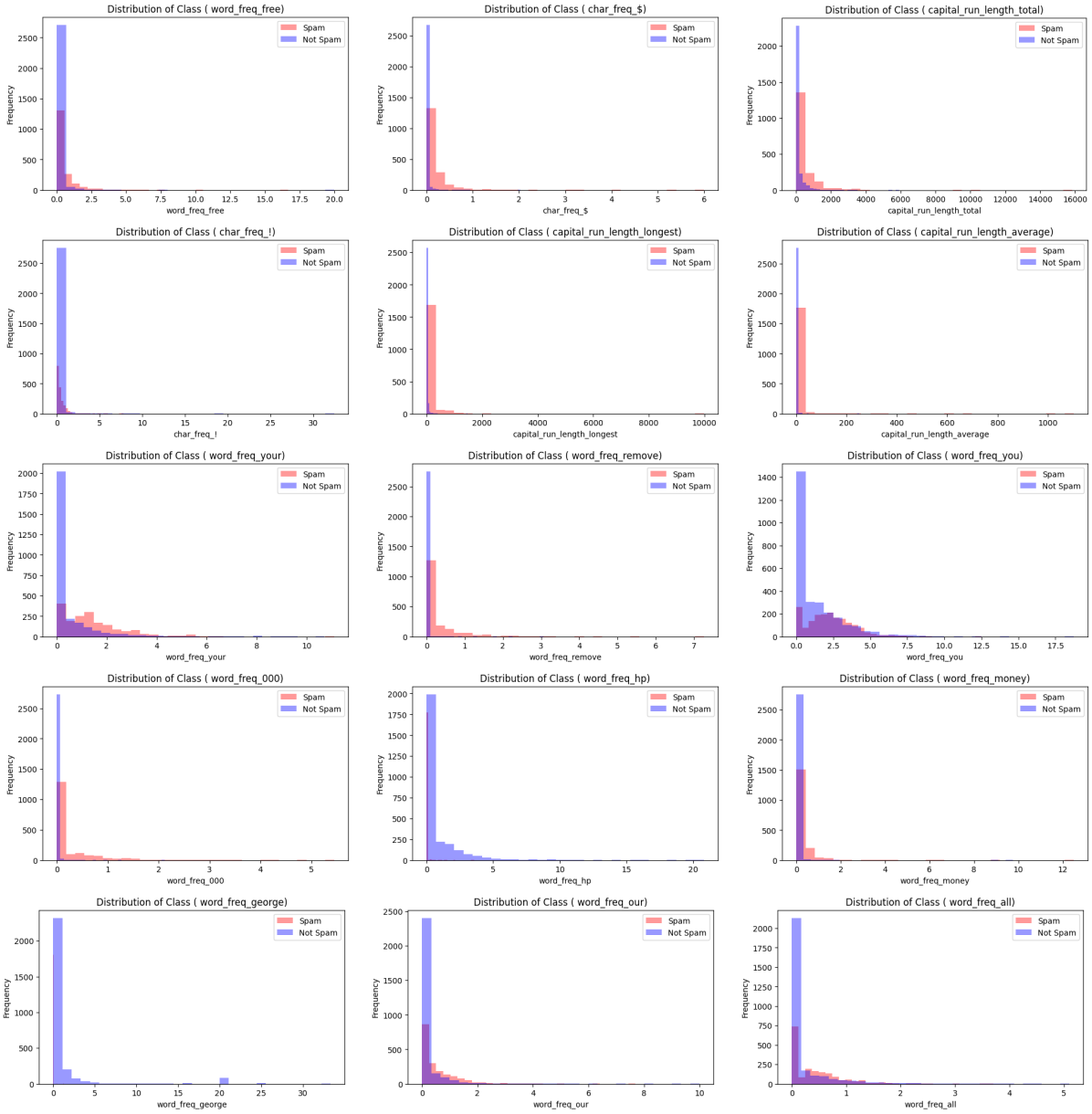

Figure 2: Comprehensive visualization of Naive Bayes results and analysis (15 plots).

Independence Violation

When a feature is duplicated, the likelihood increases multiplicatively, doubling its log-contribution. Naive Bayes thus overweights its importance, potentially distorting classifier confidence.

6. Conclusion

The Naive Bayes classifier offers a fast and effective method for spam detection. A smoothed Gaussian model with $\alpha = 0.001$ provides the best generalization. However, correlated features can violate the independence assumption, affecting model reliability. Hyperparameter tuning and metric analysis are essential for robust deployment.