# Binary Classification using Naïve Bayes and K-Nearest Neighbors on the Spambase Dataset

Ramlath Nisha A
Department of Computer Science and Engineering
Sri Sivasubramaniya Nadar College of Engineering, Chennai
Email: ramlathnisha2310611@ssn.edu.in

***Abstract*—This experiment investigates the performance of Naïve Bayes and K-Nearest Neighbors (KNN) classifiers on a benchmark binary classification problem derived from the Spambase dataset. After applying appropriate preprocessing steps such as handling missing values and feature scaling, multiple Naïve Bayes variants (Gaussian, Multinomial, and Bernoulli) and KNN models are trained and evaluated. KNN hyperparameters are tuned using cross-validation, and KDTree and BallTree neighbor search methods are compared. The models are assessed using accuracy, precision, recall, F1 score, specificity, false positive rate, and computational time. Furthermore, we analyze overfitting, underfitting, and the bias–variance characteristics of Naïve Bayes and KNN.**

***Index Terms*—Naïve Bayes, K-Nearest Neighbors, Spambase, Binary Classification, Hyperparameter Tuning, KDTree, BallTree, Bias–Variance Trade-off**

## I. Introduction

Binary classification is a fundamental task in machine learning, with applications such as email spam detection, fraud detection, and medical diagnosis. In this experiment, we apply Naïve Bayes and K-Nearest Neighbors (KNN) classifiers to the Spambase dataset, which distinguishes spam from non-spam (ham) emails using numerical features extracted from email content.

Naïve Bayes is a family of probabilistic classifiers based on Bayes' theorem with strong (naïve) independence assumptions between features. It is simple, computationally efficient, and particularly effective on high-dimensional data. Different variants such as Gaussian, Multinomial, and Bernoulli Naïve Bayes are suitable for different types of input features.

KNN is a non-parametric, instance-based learning algorithm that classifies a sample based on the labels of its nearest neighbors in the feature space. Its performance heavily depends on the choice of the number of neighbors ($k$), the distance metric, and the neighbor search strategy. Feature scaling is crucial for distance-based methods to avoid dominance of features with larger scales.

This report describes the entire workflow: preprocessing, model training, hyperparameter tuning using GridSearchCV or RandomizedSearchCV, comparison of KDTree and BallTree neighbor search methods, and analysis of overfitting, underfitting, and the bias–variance trade-off for Naïve Bayes and KNN.

## II. Aim and Objective

The aim of this experiment is to implement and compare Naïve Bayes and KNN classifiers on a binary classification dataset and to analyze their performance, generalization capability, and computational efficiency.

The specific objectives are:

- To implement Naïve Bayes (Gaussian, Multinomial, Bernoulli) and KNN classifiers for a binary classification problem.
- To preprocess the dataset by handling missing values and applying feature scaling.
- To perform EDA and visualize class distribution and feature behavior.
- To tune KNN hyperparameters (e.g., $k$, distance metric, weights, neighbor search algorithm) using GridSearchCV or RandomizedSearchCV with 5-fold cross-validation.
- To compare KDTree and BallTree neighbor search strategies in terms of accuracy and computational time.
- To evaluate all models using accuracy, precision, recall, F1 score, specificity, false positive rate, training time, and prediction time.
- To analyze overfitting, underfitting, and the bias–variance characteristics of Naïve Bayes and KNN.

## III. Dataset Description

The dataset used in this experiment is a benchmark binary classification dataset related to email spam detection.

### A. Source and Nature of Data

- Dataset reference: Kaggle – Spambase Dataset.
- Task: Binary classification (spam vs. non-spam).
- Target variable: Class label indicating whether the email is spam (1) or ham (0).

### B. Features

The dataset consists of numerical features extracted from email content, for example:

- Frequency of specific words or characters in the email (e.g., "free", "money", "!" etc.).
- Statistical measures such as average length of uninterrupted capital letters.

You may specify:

- Total number of samples: XXX

- Number of features: XXX
- Class distribution: number/percentage of spam vs. non-spam emails.

## IV. Preprocessing Steps

Preprocessing is essential for obtaining reliable performance from Naïve Bayes and KNN.

### A. Handling Missing Values

- Check for missing values in the dataset.
- If present, impute numerical features using appropriate strategies (e.g., mean or median).

### B. Feature Scaling

Since KNN is distance-based, feature scaling is applied:

- Standardization of features to zero mean and unit variance using, for example, `StandardScaler`.

### C. Train–Test Split

- The dataset is split into training and testing sets.
- Typical split: $X\%$ training and $Y\%$ testing (e.g., 80–20).

## V. Implementation Details

The experiment is implemented in Python using the scikit-learn framework.

### A. Models Implemented

- **Gaussian Naïve Bayes**: Assumes continuous features follow a Gaussian distribution.
- **Multinomial Naïve Bayes**: Suitable for count features (e.g., word counts).
- **Bernoulli Naïve Bayes**: Suitable for binary features (e.g., presence/absence of a word).
- **K-Nearest Neighbors (KNN)**: Instance-based classifier using $k$ nearest neighbors in feature space.

### B. KNN Hyperparameter Search Space

KNN hyperparameters are tuned using GridSearchCV or RandomizedSearchCV with 5-fold cross-validation. A typical search space includes:

- Number of neighbors $k$: e.g., $\{1, 3, 5, 7, 9, 11, 15\}$.
- Weights: `uniform`, `distance`.
- Distance metric: e.g., Euclidean (Minkowski with $p = 2$).
- Neighbor search algorithm: `kd_tree`, `ball_tree`, `brute`.

### C. Neighbor Search Methods

- **KDTree**: Efficient for lower-dimensional data; partitions the space using k-d trees.
- **BallTree**: Better suited for higher-dimensional spaces; partitions using hyperspheres.

For both KDTree and BallTree, the classification accuracy is primarily determined by $k$ and the distance metric; the main differences are in training and prediction time and memory usage.

### D. Evaluation Metrics

The following metrics are computed for Naïve Bayes and KNN:

- Accuracy
- Precision
- Recall
- F1 Score
- Specificity
- False Positive Rate (FPR)
- Training Time
- Prediction Time

## VI. Visualizations

This section describes the visualizations required by the assignment, with placeholders for the figures.

### A. Class Distribution Plot

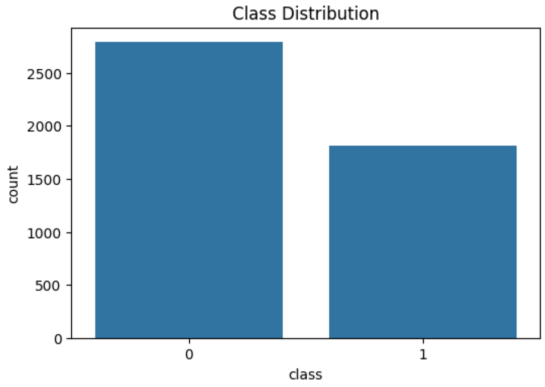


Fig. 1. Class distribution of spam and non-spam emails.

### B. Feature Distribution Plots

Fig. 2. Distribution of an example feature in the dataset.

### C. ROC Curves for Each Classifier

### D. Accuracy vs. k Plot for KNN

### E. Training vs. Validation Accuracy Plot

## VII. Performance Tables

This section contains all the tables specified in the assignment.

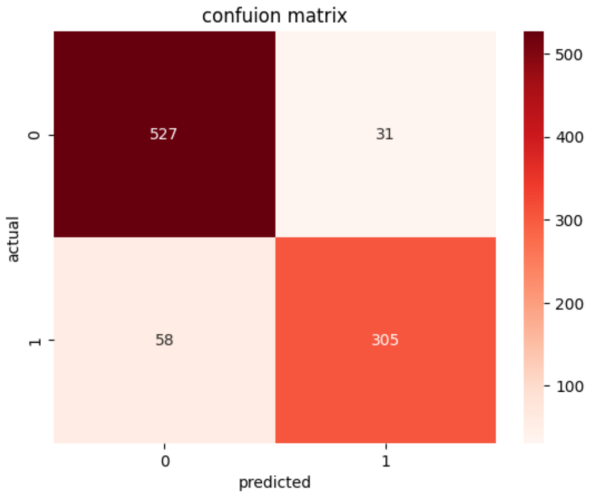


Fig. 3. Confusion matrix for Bernoulli Naïve Bayes.

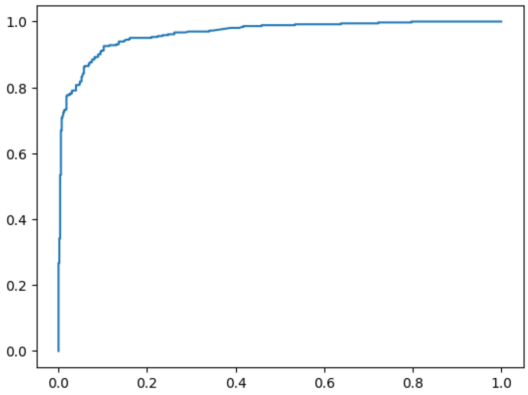


Fig. 4. ROC curves for Naïve Bayes variants and KNN.

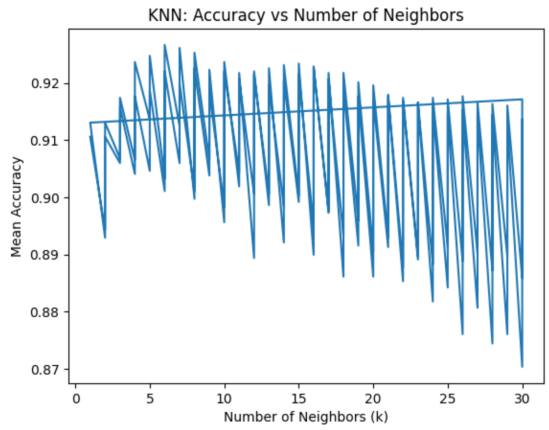


Fig. 5. Accuracy vs. number of neighbors $k$ for KNN.

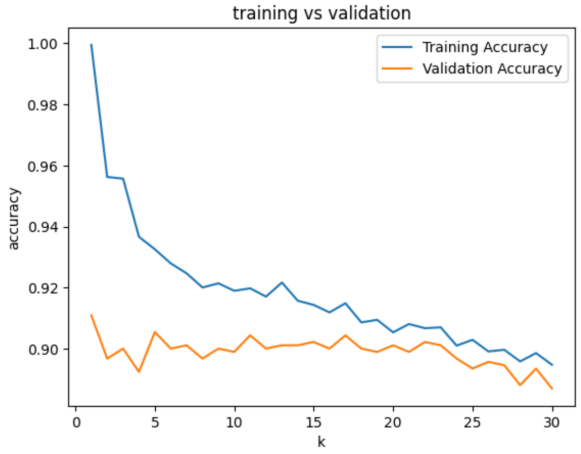


Fig. 6. Training vs. validation accuracy across different values of $k$.

TABLE I
NAÏVE BAYES PERFORMANCE METRICS

| Metric | Gaussian NB | Multinomial NB | Bernoulli NB |
|---|---|---|---|
| Accuracy | 0.83 | 0.86 | 0.90 |
| -Precision | 0.71 | 0.85 | 0.90 |
| Recall | 0.95 | 0.79 | 0.84 |
| F1 Score | 0.82 | 0.82 | 0.87 |
| Specificity | 0.75 | 0.90 | 0.94 |
| Training Time (s) | 0.008 | 0.01 | 0.01 |

### *A. Naïve Bayes Performance Metrics*

### *B. KNN Hyperparameter Tuning*

### *C. KNN Performance using KDTree*

### *D. KNN Performance using BallTree*

## VIII. OVERFITTING AND UNDERFITTING ANALYSIS

This section discusses the relationship between training and validation accuracy, the effect of $k$, and the role of hyperparameter tuning.

TABLE II
KNN HYPERPARAMETER TUNING

| Search Method | Best $k$ | Best CV Accuracy | |
|---|---|---|---|
| Grid Search | 6 | 0.9004 | {'metric': 'manhattan', |
| Randomized Search | 11 | 0.8981 | {'weights': 'distance', |

TABLE III
KNN PERFORMANCE USING KDTREE

| Metric | Value |
|---|---|
| Optimal $k$ | 6 |
| Accuracy | 0.92 |
| Precision | 0.93 |
| Recall | 0.88 |
| F1 Score | 0.90 |
| Training Time (s) | 0.02 |
| Prediction Time (s) | 0.32 |

TABLE IV
KNN PERFORMANCE USING BALLTREE

| Metric | Value |
|---|---|
| Optimal $k$ | 6 |
| Accuracy | 0.92 |
| Precision | 0.93 |
| Recall | 0.88 |
| F1 Score | 0.90 |
| Training Time (s) | 0.01 |
| Prediction Time (s) | 0.29 |

### A. Training vs. Validation Accuracy

- Compare training and validation accuracies for KNN across different values of $k$.
- Highlight cases where training accuracy is high but validation accuracy is lower, indicating overfitting.
- Identify values of $k$ where training and validation accuracies are both relatively low, indicating underfitting.

### B. Effect of Small and Large Values of $k$

- Discuss how very small $k$ (e.g., $k = 1$) leads to low bias but high variance and can overfit noise.
- Explain how very large $k$ leads to smoother decision boundaries, higher bias, and possible underfitting.

### C. Role of Hyperparameter Tuning in Generalization

- Describe how GridSearchCV or RandomizedSearchCV helps find an optimal value of $k$ that balances bias and variance.
- Relate the chosen $k$ and other tuned parameters to improvements in validation accuracy and test performance.

## IX. BIAS–VARIANCE ANALYSIS

This section focuses on bias and variance behavior of Naïve Bayes and KNN.

### A. Bias Behavior of Naïve Bayes

- Explain that Naïve Bayes makes strong independence assumptions among features, which can introduce bias.
- Despite the bias, Naïve Bayes can generalize well on high-dimensional data when the assumptions are approximately satisfied.

### B. Variance Behavior of KNN

- Discuss how KNN is a low-bias, high-variance model, especially for small values of $k$.
- Explain how increasing $k$ smooths the decision boundary, reducing variance but increasing bias.

### C. Effect of Tuning on Bias–Variance Trade-off

- Describe how tuning $k$ and other KNN hyperparameters adjusts the bias–variance trade-off.
- Compare the final tuned KNN model with Naïve Bayes in terms of bias, variance, and generalization performance.

## X. OBSERVATIONS AND CONCLUSION

### A. Key Observations

- Summarize the performance of Gaussian, Multinomial, and Bernoulli Naïve Bayes in terms of accuracy, precision, recall, F1 score, and specificity.
- Highlight which Naïve Bayes variant worked best for the Spambase dataset and why (e.g., suitability of feature type).
- Summarize the effect of KNN hyperparameter tuning on cross-validation and test performance.
- Compare KDTree and BallTree in terms of accuracy, training time, prediction time, and memory usage.

### B. Conclusion

In this experiment, Naïve Bayes and KNN classifiers were implemented and evaluated on the Spambase dataset. Naïve Bayes provided a simple probabilistic baseline with relatively fast training and prediction times. KNN, after proper feature scaling and hyperparameter tuning, achieved competitive or superior accuracy, at the cost of higher computational requirements during prediction.

The analysis of training and validation accuracies across different values of $k$ highlighted clear overfitting and underfitting regimes. Hyperparameter tuning using 5-fold cross-validation helped identify an optimal configuration that balanced bias and variance. The comparison between KDTree and BallTree showed that both methods yielded similar classification accuracy, with differences mainly in computational efficiency and memory usage.

Overall, the experiment demonstrated the strengths and limitations of Naïve Bayes and KNN for binary classification, emphasizing the importance of preprocessing, hyperparameter tuning, and a careful analysis of the bias–variance trade-off.