**Sri Sivasubramaniya Nadar College of Engineering, Chennai**
(An autonomous Institution affiliated to Anna University)

| | | |
|---|---|---|
| Degree & Branch | B.E. Computer Science & Engineering | Semester VI |
| Subject Code & Name | UCS2612 – Machine Learning Algorithms Laboratory | |
| Academic Year | 2025–2026 (Even) | Batch 2023–2027 |
| Due Date | **26/01/2026** | |

R Padmashri
3122 23 5001 093

**Experiment 2: Binary Classification using Naïve Bayes and K-Nearest Neighbors**

# Objective

To implement Naïve Bayes and K-Nearest Neighbors (KNN) classifiers for a binary classification problem, evaluate them using multiple performance metrics, visualize model behavior, and analyze bias–variance characteristics.

# Dataset

The Spambase dataset is used for binary email spam classification. It consists of 4601 instances with 57 numerical features.

Dataset reference:

- Kaggle: Spambase Dataset

# Exploratory Data Analysis

## 0.1 Required Imports

Listing 1: Importing required libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import time

from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.naive_bayes import GaussianNB, MultinomialNB, BernoulliNB
from sklearn.neighbors import KNeighborsClassifier

from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
    confusion_matrix, roc_curve, auc
```

### Class Distribution

The class distribution plot shows the proportion of spam and non-spam emails in the dataset. The dataset exhibits a moderately balanced distribution, ensuring that the classification models are trained without significant class imbalance bias.

Listing 2: Class distribution visualization

```
sns.countplot(x=y)
plt.title("Class Distribution")
plt.xlabel("Class Label")
plt.ylabel("Count")
plt.show()
```

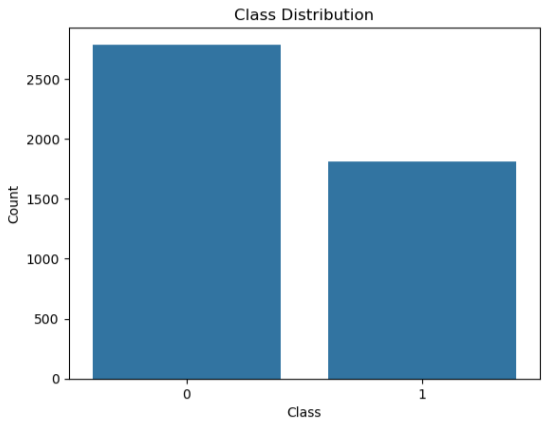


Figure 1: Class distribution of spam and non-spam emails

## Naïve Bayes Classification

Three Naïve Bayes variants were evaluated: Gaussian Naïve Bayes, Multinomial Naïve Bayes, and Bernoulli Naïve Bayes. Models were tested using different train–test split ratios.

Listing 3: Naive Bayes models

```
models = {
    "Gaussian NB": GaussianNB(),
    "Multinomial NB": MultinomialNB(),
    "Bernoulli NB": BernoulliNB()
}

for name, model in models.items():
    start = time.time()
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    end = time.time()
```

## Detailed Comparison (25% Test Split)

Table 1: Naïve Bayes Performance Metrics

| Metric | Gaussian NB | Multinomial NB | Bernoulli NB |
|---|---|---|---|
| Accuracy | 0.8306 | 0.7767 | 0.8818 |
| Precision | 0.7133 | 0.7244 | 0.8768 |
| Recall | 0.9537 | 0.7004 | 0.8150 |
| F1 Score | 0.8162 | 0.7122 | 0.8447 |
| Specificity | 0.7504 | 0.8264 | 0.9254 |
| Training Time (s) | 0.0049 | 0.0150 | 0.0082 |

## Detailed Comparison (30% Test Split)

Table 2: Naïve Bayes Performance Metrics (30% Test Split)

| Metric | Gaussian NB | Multinomial NB | Bernoulli NB |
|---|---|---|---|
| Accuracy | 0.8226 | 0.8067 | 0.8986 |
| Precision | 0.7023 | 0.7560 | 0.8992 |
| Recall | 0.9540 | 0.7518 | 0.8364 |
| F1 Score | 0.8090 | 0.7539 | 0.8667 |
| Specificity | 0.7372 | 0.8423 | 0.9391 |
| Training Time (s) | 0.0051 | 0.0037 | 0.0055 |

## Detailed Comparison (18% Test Split)

Table 3: Naïve Bayes Performance Metrics (18% Test Split)

| Metric | Gaussian NB | Multinomial NB | Bernoulli NB |
|---|---|---|---|
| Accuracy | 0.8166 | 0.7575 | 0.8987 |
| Precision | 0.6923 | 0.6875 | 0.8907 |
| Recall | 0.9633 | 0.7064 | 0.8471 |
| F1 Score | 0.8056 | 0.6968 | 0.8683 |
| Specificity | 0.7211 | 0.7908 | 0.9323 |
| Training Time (s) | 0.0057 | 0.0044 | 0.0068 |

### Performance for Different Test Splits

Table 4: Naïve Bayes Performance for Different Test Splits

| Test Split | Best Model | Accuracy | F1 Score |
|---|---|---|---|
| 25% | Bernoulli NB | 0.8818 | 0.8447 |
| 30% | Bernoulli NB | 0.8986 | 0.8667 |
| 18% | Bernoulli NB | 0.8987 | 0.8683 |

Table 5: Inference on Test Split Variations for Naïve Bayes

| Test Split | Best Model | Accuracy | Inference |
|---|---|---|---|
| 25% | Bernoulli NB | 0.8818 | Moderate training data resulted in good generalization. |
| 30% | Bernoulli NB | 0.8986 | Slightly larger test set improved robustness of evaluation. |
| 18% | Bernoulli NB | 0.8987 | More training data led to the highest and most stable performance. |

## K-Nearest Neighbors Classification

### Baseline KNN Performance

A baseline K-Nearest Neighbors (KNN) classifier was initially implemented using default parameters to understand the effect of the number of neighbors on classification performance. The model was evaluated for different values of k, and the accuracy trend was analyzed using an accuracy versus k plot to identify potential overfitting and underfitting regions.

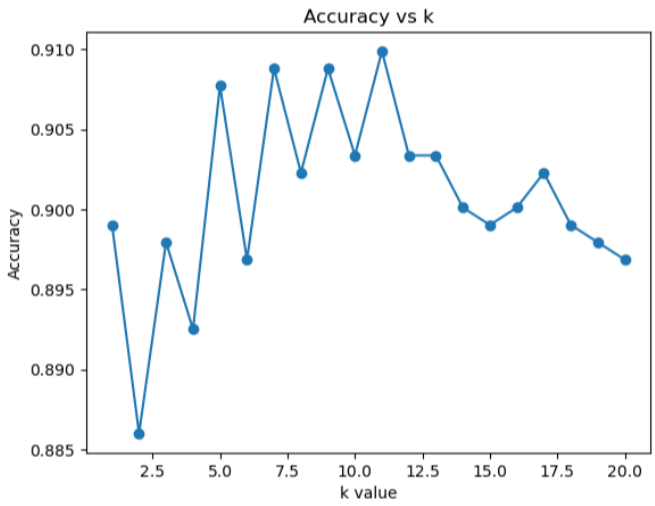


Figure 2: Accuracy vs number of neighbors ($k$) for KNN

### Hyperparameter Tuning Results

Hyperparameter tuning was performed on the KNN classifier using GridSearchCV and RandomizedSearchCV to identify the optimal combination of parameters. Both methods explored different values of the number of neighbors, distance metrics, and weighting schemes using cross-validation. Grid Search provided the best overall performance with a slightly higher validation accuracy, while Randomized Search achieved comparable results with reduced computational cost.

Table 6: KNN Hyperparameter Tuning

| Search Method | Best $k$ | Best CV Accuracy | Best Parameters |
|---|---|---|---|
| Grid Search | 10 | 0.9253 | Manhattan, Distance |
| Randomized Search | 16 | 0.9204 | Euclidean, Distance |

### Optimized KNN Performance

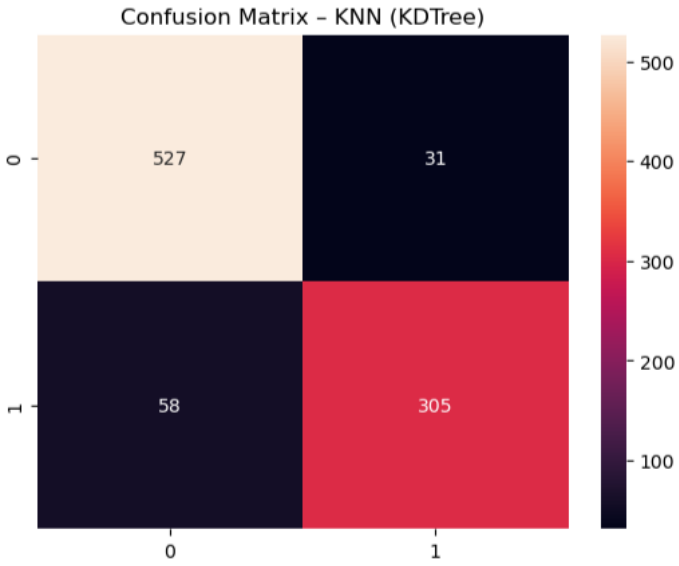


Figure 3: Confusion matrix for optimized KNN classifier

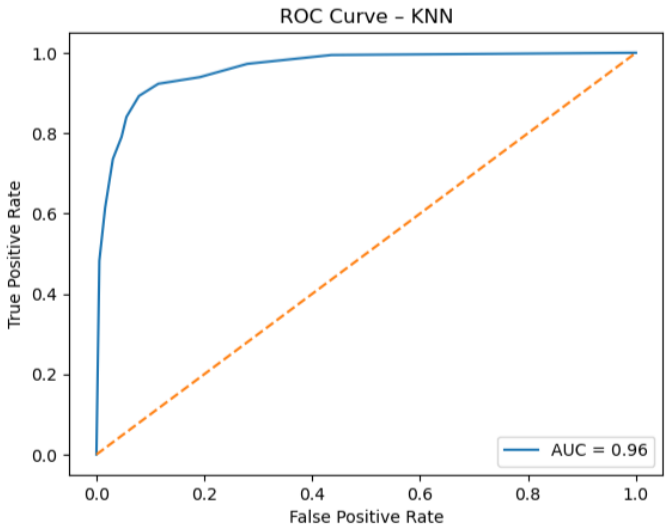


Figure 4: ROC curve for optimized KNN classifier

## KDTree vs BallTree Comparison

Table 7: Comparison of Neighbor Search Algorithms

| Criterion | KDTree | BallTree |
|---|---|---|
| Accuracy | Similar | Similar |
| Training Time | Faster | Slower |
| Prediction Time | Faster | Moderate |
| Memory Usage | Low | Medium |

## Bias–Variance Analysis

Naïve Bayes exhibits higher bias due to its strong independence assumption. KNN shows higher variance for small values of $k$ and improved generalization after hyperparameter tuning.

## Conclusion

Bernoulli Naïve Bayes achieved the best performance among Naïve Bayes variants. KNN performance improved significantly after hyperparameter tuning, with KDTree providing faster execution. Overall, tuning and model selection improved generalization and classification accuracy.

## References


- Scikit-learn: Naïve Bayes
- Scikit-learn: KNN
- Scikit-learn: Hyperparameter Optimization
- Spambase Dataset