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
Name:	Piranow c	Roll No:	3122235001096
Due Date	27-01-2026		

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

1. Aim & 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 overfitting, underfitting, and bias-variance characteristics.

2. Dataset

The experiment uses the **Spambase dataset**, a benchmark binary classification dataset designed for spam email detection. The dataset consists of numerical features extracted from email content and a binary class label indicating whether an email is *spam* or *non-spam*.

Each instance in the dataset represents a single email described using frequency-based features, making it suitable for evaluating probabilistic and distance-based classifiers such as Naïve Bayes and K-Nearest Neighbors.

Dataset reference:

- Kaggle: Spambase Dataset

3. Preprocessing Steps

- The dataset file `spambase_csv_Kaggle.csv` was loaded into a Pandas DataFrame for analysis.
- The dataset was separated into the feature matrix (X), containing all numerical attributes, and the target vector (y), representing the spam and non-spam class labels.
- Feature scaling was performed using `StandardScaler` from the Scikit-learn library to normalize the input features, which is essential for distance-based algorithms such as KNN.
- The normalized data was split into training and testing sets using a standard train–test split strategy to evaluate model generalization performance.
- The resulting subsets X_{train}, X_{test}, y_{train}, and y_{test} were used for model training and evaluation.

4. Implementation Details

- Implemented Gaussian Naive Bayes, Bernoulli's Naive Bayes and Multinomial Naive Bayes.
- Implemented baseline K-Nearest Neighbors (KNN) Algorithm and with both KD-Tree and Ball-Tree Algorithm.
- Implemented hyperparameter tuning for KNN-Algorithm using the GridSearchCV & RandomizedSearchCV.
- Compared all the algorithm running time and its accuracy score and plotted the various plots to visualize the algorithm implementation.

5. Visualizations

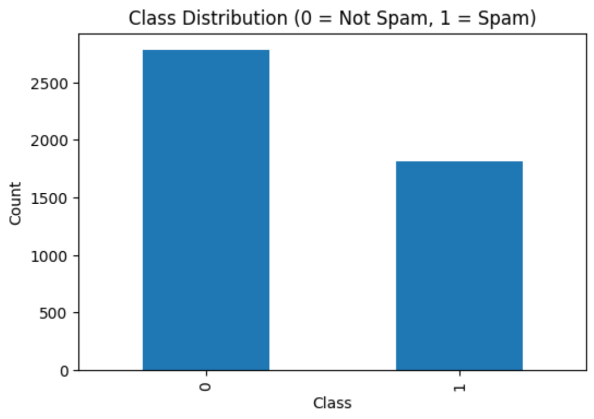

Figure 1: Visualization 1

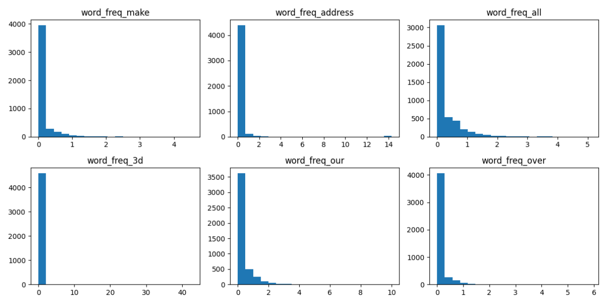

Figure 2: Visualization 2

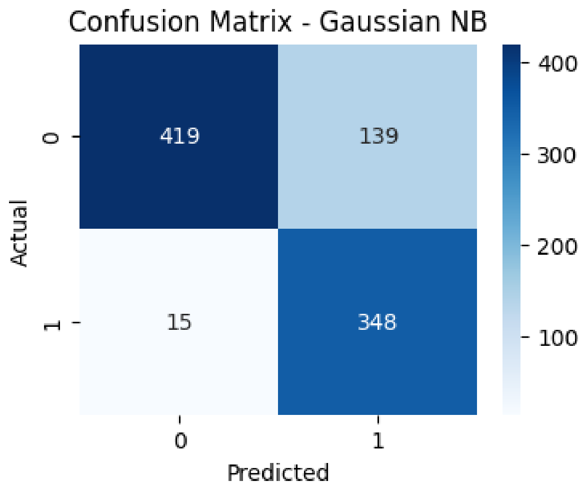

Figure 3: Visualization 3

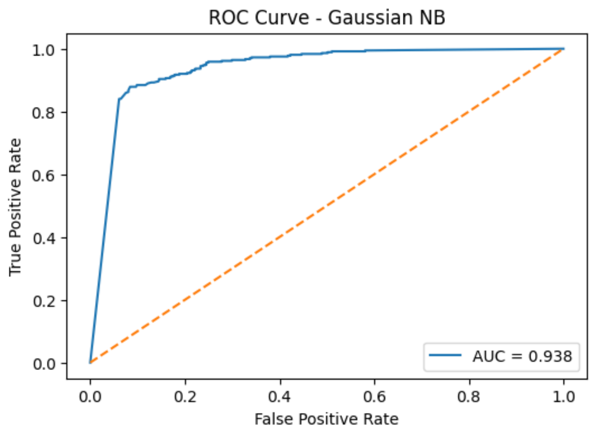

Figure 4: Visualization 4

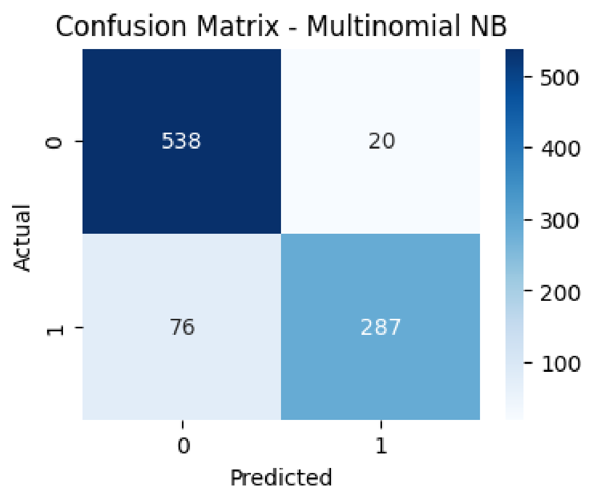

Figure 5: Visualization 5

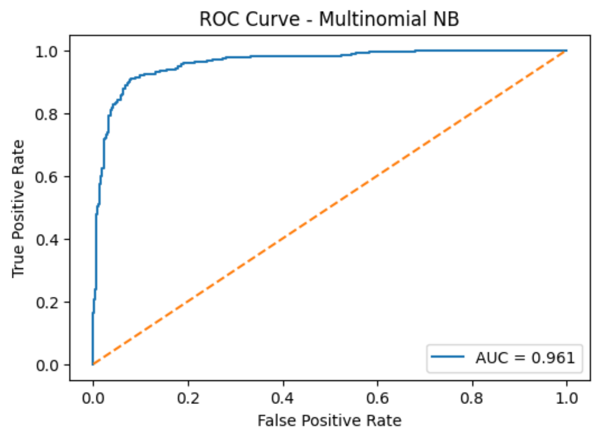

Figure 6: Visualization 6

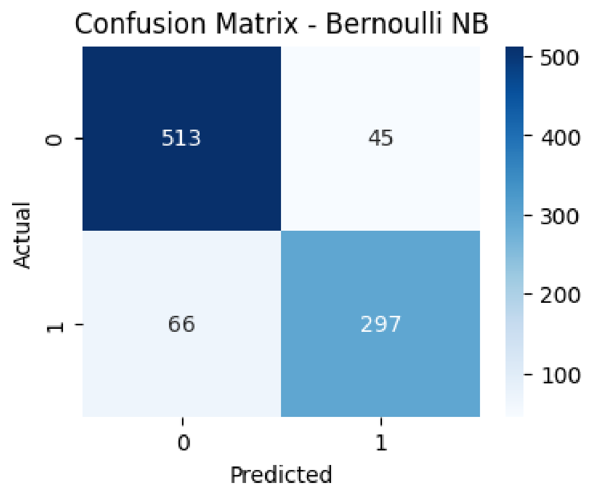

Figure 7: Visualization 7

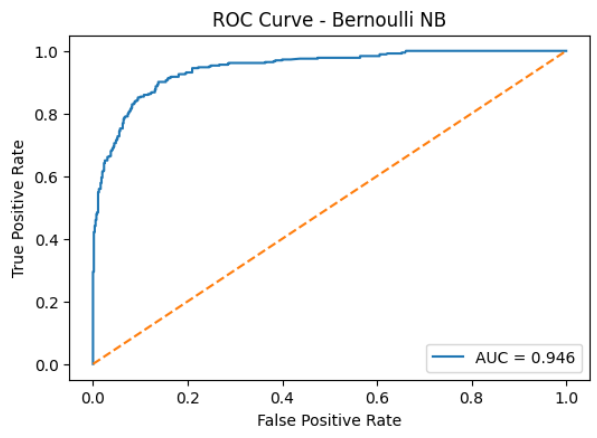

Figure 8: Visualization 8

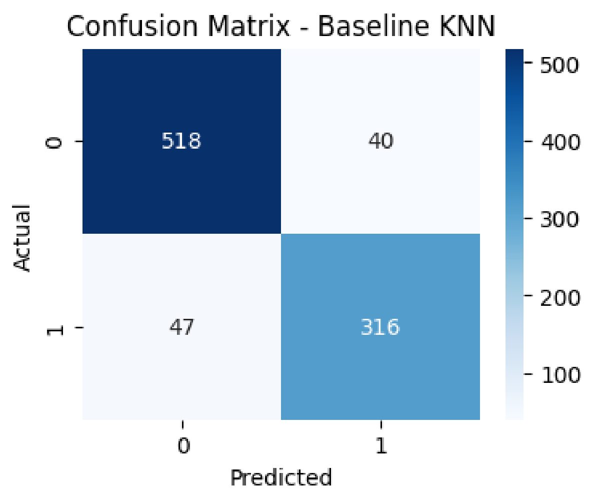

Figure 9: Visualization 9

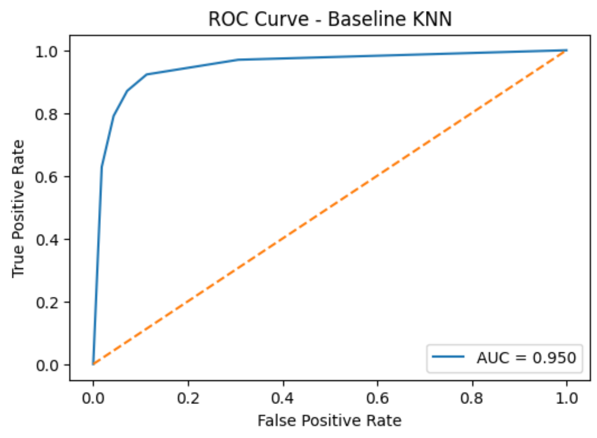

Figure 10: Visualization 10

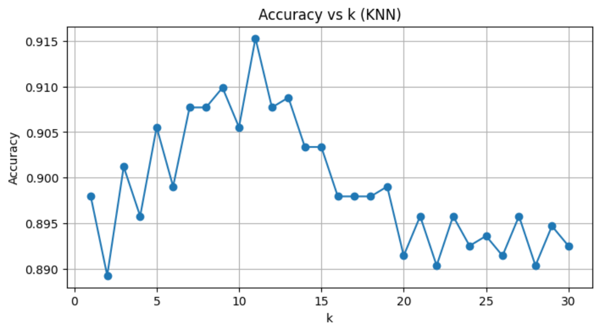

Figure 11: Visualization 11

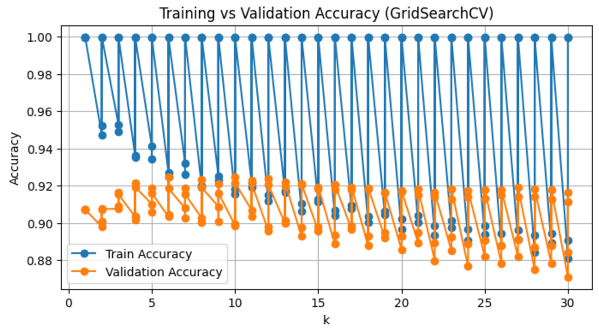

Figure 12: Visualization 12

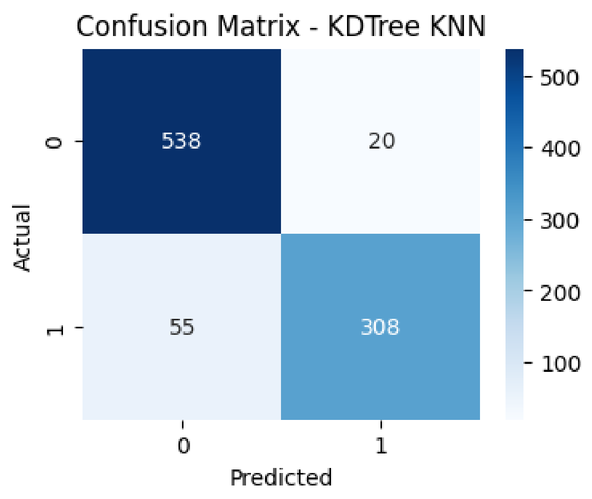

Figure 13: Visualization 13

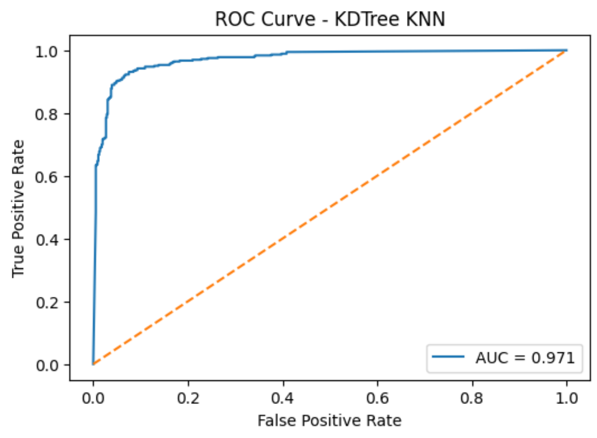

Figure 14: Visualization 14

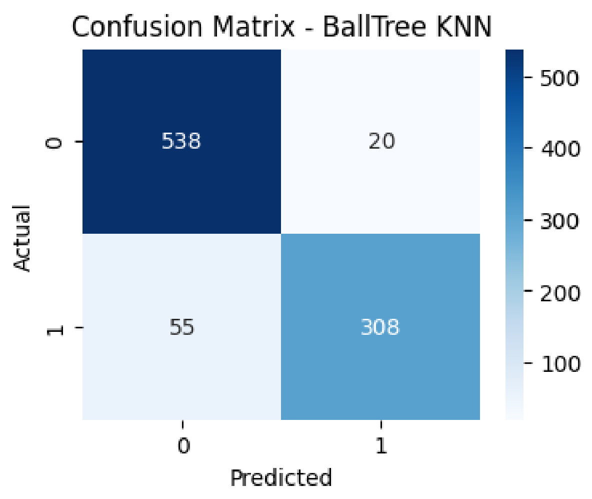

Figure 15: Visualization 15

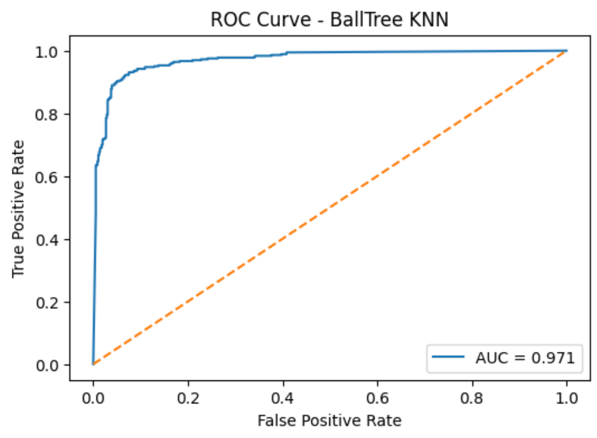

Figure 16: Visualization 16

graphicx float

6. Performance Comparison

Table 1: Naïve Bayes Performance Metrics

Metric	**Gaussian NB**	**Multinomial NB**	**Bernoulli NB**
Accuracy	0.832790	0.895765	0.879479
Precision	0.714579	0.934853	0.868421
Recall	0.958678	0.790634	0.818182
F1 Score	0.818824	0.856716	0.842553
Specificity	0.750896	0.964158	0.919355
False Positive Rate	0.249104	0.035842	0.080645
Training Time (s)	0.018870	0.033318	0.059150
Prediction Time (s)	0.004327	0.002874	0.006165

Table 2: KNN Hyperparameter Tuning

Search Method	**Best k**	**Best CV Accuracy**	**Best Parameters**
Grid Search	10	0.924728	–
Randomized Search	14	0.920924	–

Table 3: KNN Performance using BallTree

Metric	**Value**
Accuracy	0.918567
Precision	0.939024
Recall	0.848485
F1 Score	0.891462
Specificity	0.964158
False Positive Rate	0.035842
Training Time (s)	0.036658
Prediction Time (s)	0.389959

Table 4: KNN Performance using KDTree

Metric	**Value**
Accuracy	0.918567
Precision	0.939024
Recall	0.848485
F1 Score	0.891462
Specificity	0.964158
False Positive Rate	0.035842
Training Time (s)	0.054384
Prediction Time (s)	0.354829

Table 5: Comparison of Neighbor Search Algorithms

Criterion	**KDTree**	**BallTree**
Accuracy	0.918567	0.918567
Training Time (s)	0.054384	0.036658
Prediction Time (s)	0.354829	0.389959
Memory Usage	Low/Medium	Medium/High

7. Overfitting and Underfitting Analysis

- **Training vs. validation accuracy behavior in KNN**
  - Experimental results show that for smaller values of k, the KNN classifier achieves very high training accuracy but relatively lower validation accuracy.
  - This gap indicates overfitting, where the model memorizes training samples and becomes sensitive to noise.
  - As the value of k increases, the difference between training and validation accuracy reduces, indicating improved generalization.
  - For optimal values of k identified using cross-validation, both training and validation accuracies are stable and closely aligned.
  - When k becomes very large, both training and validation accuracies decrease, indicating underfitting due to overly smooth decision boundaries.
- **Effect of neighborhood size (k)**
  - **Small k values:**
    - High training accuracy and lower validation accuracy
    - High variance and sensitivity to outliers
    - Indicates overfitting
  - **Moderate k values (selected via tuning):**
    - Balanced training and validation accuracy
    - Best predictive performance on unseen test data
    - Represents an optimal bias–variance trade-off

- **Large k values:**
  - Reduced accuracy on both training and validation sets
  - Loss of local structure in the data
  - Indicates underfitting

- **Impact of hyperparameter tuning**
  - Grid Search and Randomized Search were used to select optimal values of k based on cross-validation accuracy.
  - Tuned KNN models demonstrated better generalization compared to manually selected k values.
  - This confirms that systematic hyperparameter tuning effectively reduces overfitting and underfitting.
  - Selecting k solely based on training accuracy would have resulted in inferior test performance.

8. Bias–Variance Analysis

- **Bias characteristics of Naïve Bayes**
  - Naïve Bayes classifiers assume conditional independence among features.
  - This assumption limits model flexibility, resulting in higher bias.
  - Experimental results show similar training and validation accuracies, indicating low variance.
  - Although stable, the model may slightly underfit when feature dependencies are present in the data.

- **Variance characteristics of KNN**
  - The variance of the KNN classifier is strongly influenced by the choice of k.
  - Small k values lead to low bias but high variance, as reflected by fluctuating validation accuracy.
  - Increasing k reduces variance and improves model stability.
  - Optimal k values obtained through tuning achieve a balanced bias–variance trade-off.

- **Effect of tuning on bias–variance trade-off**
  - Hyperparameter tuning adjusts model complexity by selecting appropriate neighborhood sizes.
  - Tuned KNN models reduce excessive variance while avoiding high bias.
  - This results in improved validation accuracy and consistent test performance.
  - The results highlight the importance of controlled tuning for achieving robust generalization.

9. Observation & Conclusion

In this experiment, multiple binary classification models were implemented and evaluated using appropriate training, validation, and test performance metrics. Naïve Bayes classifiers demonstrated stable and consistent performance with low variance due to their strong conditional independence assumption. However, this assumption resulted in higher bias, limiting their ability to model complex feature relationships present in the dataset.

The K-Nearest Neighbors (KNN) classifier exhibited greater flexibility and achieved higher accuracy when the neighborhood size was properly tuned. Experimental results showed that small values of k led to overfitting due to high variance, while very large values of k caused underfitting. Hyperparameter tuning using Grid Search and Randomized Search successfully identified optimal values of k, improving generalization performance and reducing overfitting.

Overall, the experiment highlights the importance of balancing bias and variance through appropriate model selection and hyperparameter tuning. Cross-validation played a crucial role in ensuring fair model comparison and selecting configurations that perform well on unseen data. to achieve robust generalization.

References

- Scikit-learn Documentation: Naïve Bayes Classifiers
- Scikit-learn Documentation: K-Nearest Neighbors
- Scikit-learn Documentation: Hyperparameter Optimization
- Kaggle: Spambase Dataset