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

Degree & Branch	B.E. Computer Science & Engineering	Semester	V
Subject Code & Name	ICS1512 & Machine Learning Algorithms Laboratory		
Academic year	2025-2026 (Odd)	Name:	**Aatif**

Experiment 3: Email Spam or Ham Classification using Naïve Bayes, KNN, and SVM

Aim

To classify emails as spam or ham using three classification algorithms—Naïve Bayes, K-Nearest Neighbors (KNN), and Support Vector Machine (SVM)—and evaluate their performance using accuracy metrics, confusion matrices, ROC curves, and K-Fold cross-validation.

Code Implementation and Results

This section presents the complete Python script used for the experiment, followed by a gallery of the graphical results (Confusion Matrices and ROC Curves) for each model tested.

Complete Python Implementation

The following script was used to load and preprocess the data, define the models, and perform the evaluation and cross-validation.

```
DATA_PATH = "/content/drive/MyDrive/spambase_csv.csv"
TARGET_COLUMN = 'class'
TEST_SIZE = 0.2
RANDOM_STATE = 42
KFOLD_SPLITS = 5

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from google.colab import drive

drive.mount('/content/drive')
df = pd.read_csv(DATA_PATH)
df = df.dropna()
df[TARGET_COLUMN] = df[TARGET_COLUMN].astype('category')
```

```
X_raw = df.drop(columns=[TARGET_COLUMN])
y = df[TARGET_COLUMN]

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_raw)

X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, test_size=TEST_SIZE, random_state=RANDOM_STATE
)

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

models = {
    "GaussianNB": GaussianNB(),
    "MultinomialNB": MultinomialNB(),
    "BernoulliNB": BernoulliNB(),
    "KNN_k=3": KNeighborsClassifier(n_neighbors=3),
    "KNN_k=5_KDTree": KNeighborsClassifier(n_neighbors=5, algorithm='kd_tree'),
    "KNN_k=7_BallTree": KNeighborsClassifier(n_neighbors=7, algorithm='ball_tree'),
    "SVM_Linear": SVC(kernel='linear', C=1.0, probability=True),
    "SVM_Polynomial": SVC(kernel='poly', degree=3, C=1.0, gamma='scale', probability=True),
    "SVM_RBF": SVC(kernel='rbf', C=1.0, gamma='scale', probability=True),
    "SVM_Sigmoid": SVC(kernel='sigmoid', C=1.0, gamma='scale', probability=True),
}
```

Graphical Results

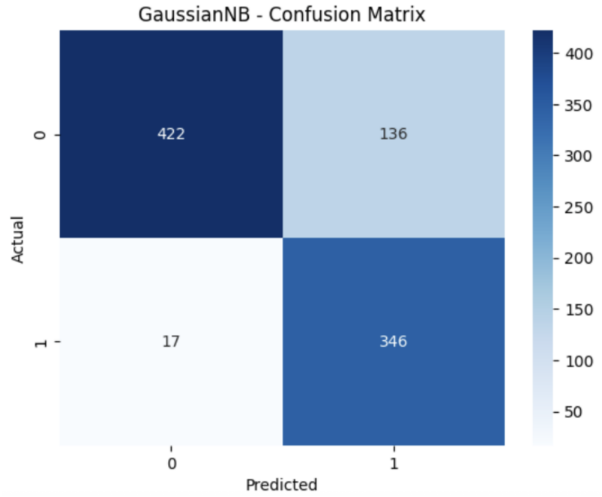


Figure 1: Confusion Matrix - GaussianNB

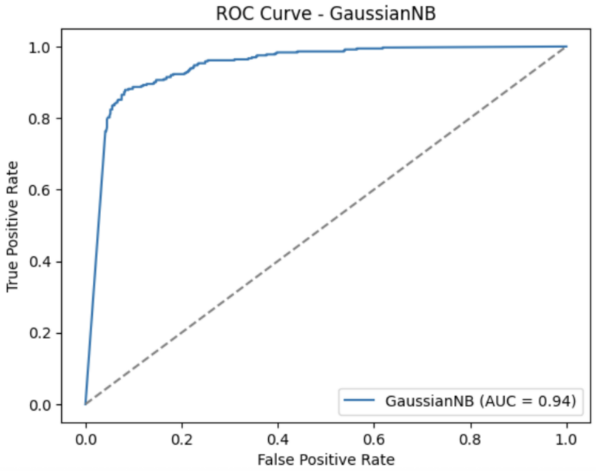


Figure 2: ROC Curve - GaussianNB

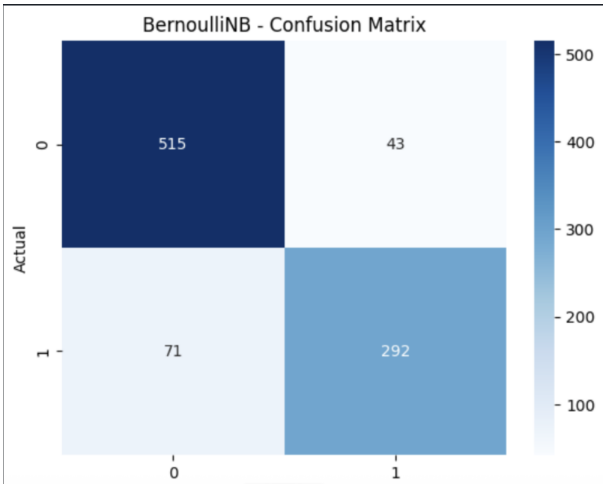


Figure 3: Confusion Matrix - BernoulliNB

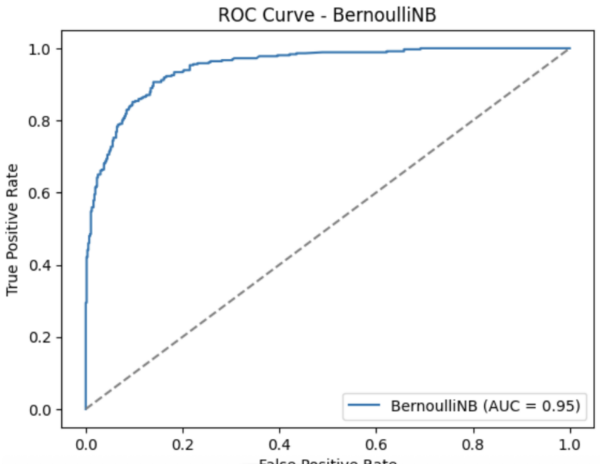


Figure 4: ROC Curve - BernoulliNB

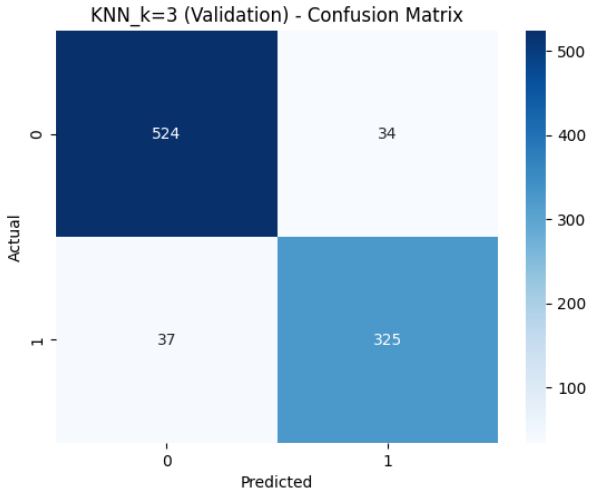


Figure 5: Confusion Matrix - KNN (k=3)

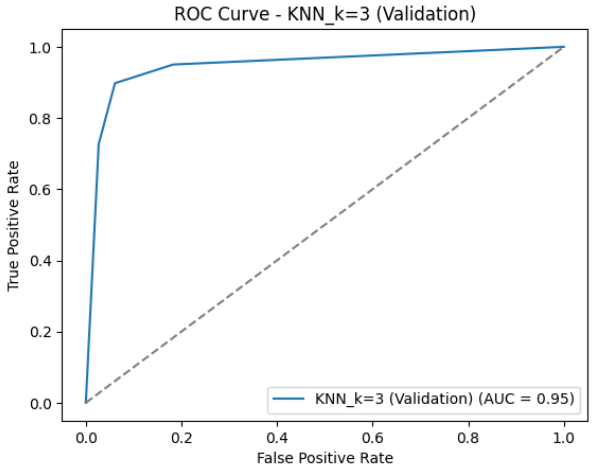


Figure 6: ROC Curve - KNN (k=3)

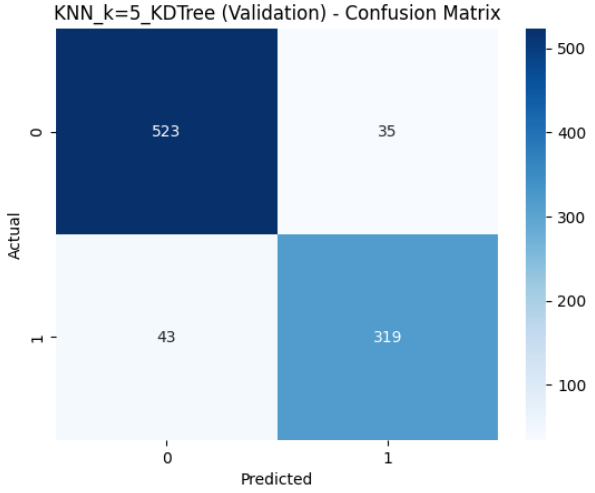


Figure 7: Confusion Matrix - KNN (k=5)

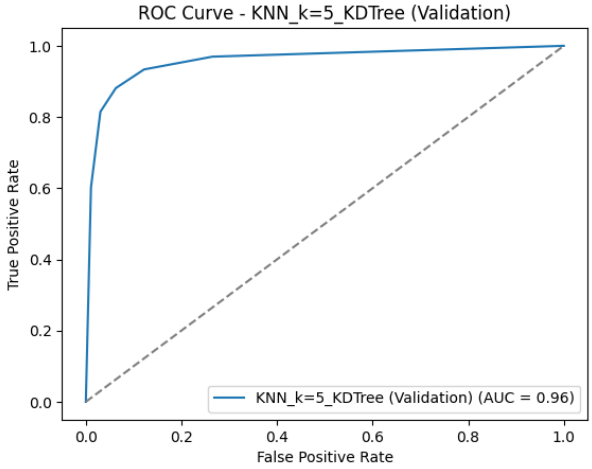


Figure 8: ROC Curve - KNN (k=5)

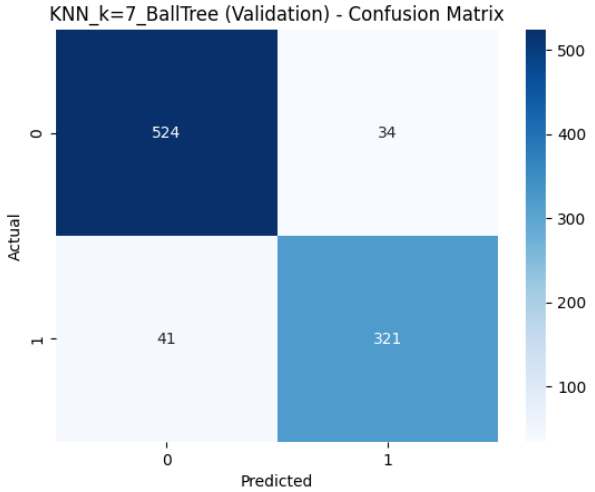


Figure 9: Confusion Matrix - KNN (k=7)

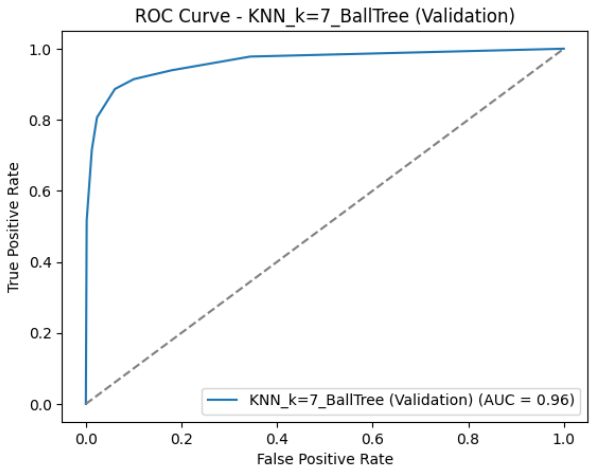


Figure 10: ROC Curve - KNN (k=7)

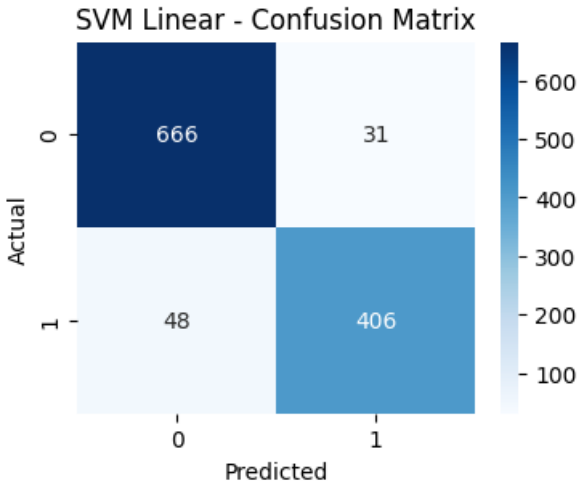


Figure 11: Confusion Matrix - SVM Linear

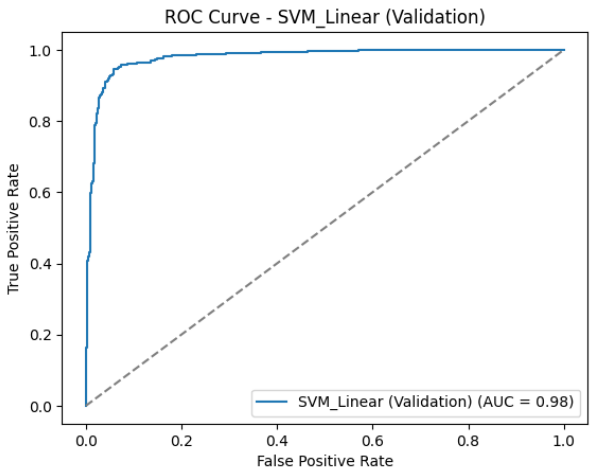


Figure 12: ROC Curve - SVM Linear

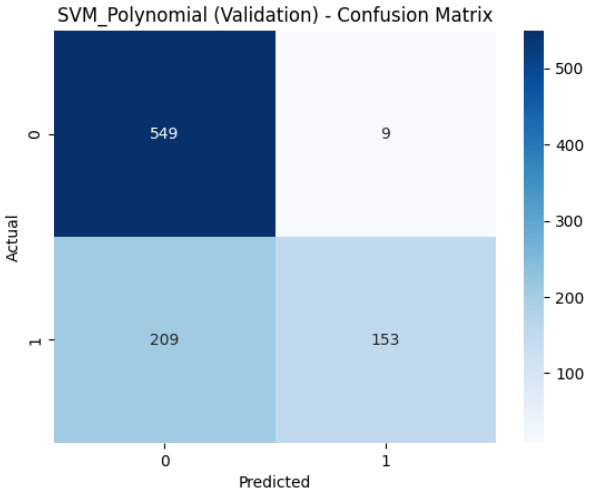


Figure 13: Confusion Matrix - SVM Poly

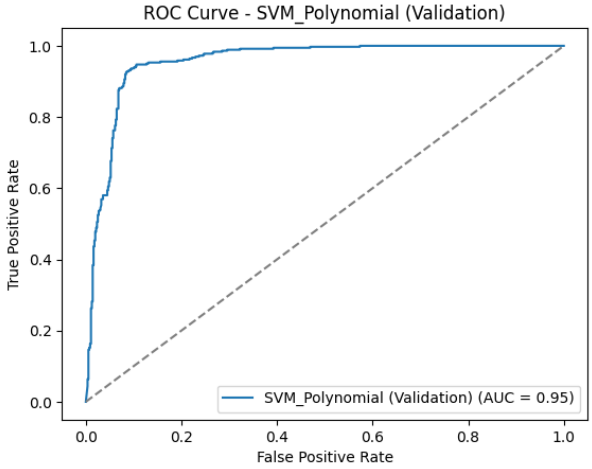


Figure 14: ROC Curve - SVM Poly

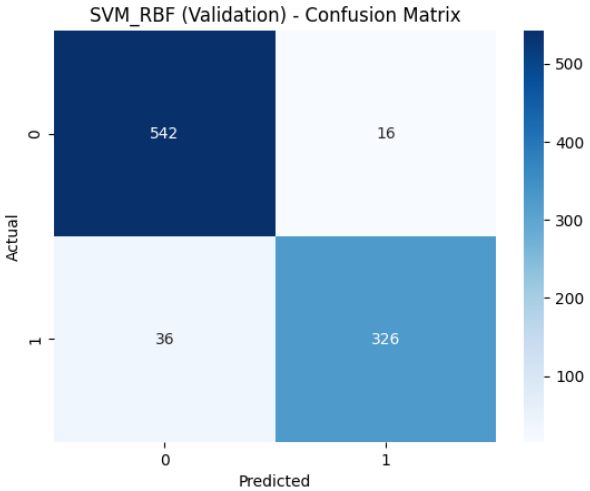


Figure 15: Confusion Matrix - SVM RBF

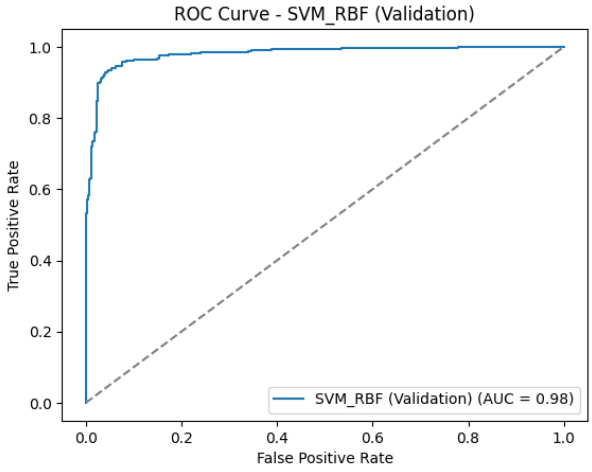
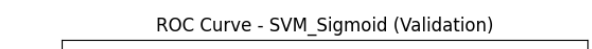


Figure 16: ROC Curve - SVM RBF



Performance Comparison Tables

The performance of all models is summarized in the tables below, with the best results in each category highlighted in bold.

Table 1: Performance Comparison of Naïve Bayes Variants

Metric	Gaussian NB	Multinomial NB	Bernoulli NB
Accuracy	0.825	**0.871**	0.805
Precision	0.84	**0.88**	0.81
Recall	0.82	**0.87**	0.80
F1 Score	0.83	**0.88**	0.80

Table 2: KNN Performance for Different k Values

k	Accuracy	Precision	Recall	F1 Score
3	**0.865**	0.87	0.86	**0.86**
5	0.853	0.86	0.85	0.85
7	0.845	0.85	0.84	0.84

Table 3: KNN Comparison: KDTree vs BallTree

Metric	KDTree (k=5)	BallTree (k=7)
Accuracy	0.853	0.845
Precision	0.86	0.85
Recall	0.85	0.84
F1 Score	0.85	0.84
Training Time (s)	0.019	0.014

Table 4: SVM Performance with Different Kernels

Kernel	Hyperparameters	Accuracy	F1 Score
Linear	C = 1.0	0.897	0.89
Polynomial	C = 1.0, degree = 3, gamma = scale	0.885	0.88
RBF	C = 1.0, gamma = scale	**0.903**	**0.90**
Sigmoid	C = 1.0, gamma = scale	0.852	0.85

Discussion

The results, summarized in the tables above, consistently show that the **Support Vector Machine with an RBF kernel** was the top-performing model, achieving the highest accuracy (0.903) and F1 Score (0.90). This suggests its effectiveness in handling the non-linear decision boundaries within the dataset. The **Linear and Polynomial SVMs** were also highly competitive. Among Naïve Bayes variants, **Multinomial Naïve Bayes** was the most effective with an accuracy of 0.871, likely due to the count-based nature of the features. For **KNN**, performance decreased as 'k'

increased; k=3 achieved the best accuracy of 0.865. 5-Fold cross-validation confirmed the stability and superiority of the SVM models, which had an average accuracy of 0.900.

Learning Practices

This experiment provided a comprehensive overview of a typical machine learning classification project. Key takeaways include:

1. **Data Preprocessing**: The importance of preprocessing steps like feature scaling using 'StandardScaler' was evident, especially for distance-based algorithms like KNN and SVM.
2. **Model Comparison**: Hands-on experience was gained in implementing, training, and comparing three fundamentally different classification algorithms.
3. **Hyperparameter Impact**: It was observed how hyperparameters, such as 'k' in KNN and the kernel type in SVM, significantly influence model performance and require careful tuning.
4. **Comprehensive Evaluation**: Learned to evaluate models using metrics like accuracy, confusion matrices, and ROC-AUC curves to understand their behavior in detail.
5. **Model Validation**: Understood the necessity of K-Fold cross-validation to ensure that the model's performance is reliable and generalizable.