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

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

Name	Nithish Ra
Reg. No.	3122237001033
Degree & Branch	M. Tech (Integrated) Computer Science & Engineering
Semester	V
Subject	ICS1512 & Machine Learning Algorithms Laboratory
Academic Year	2025–2026 (Odd)
Batch	2023–2028

Contents

1 Aim and Objective

The aim of this experiment is to classify emails as either 'spam' or 'ham' (not spam) using three distinct classification algorithms: K-Nearest Neighbors (KNN), Naïve Bayes, and Support Vector Machine (SVM). The primary objective is to build, train, and evaluate models for each algorithm, compare their performance using metrics like accuracy, precision, recall, and F1-score, and use K-Fold cross-validation to assess their generalization capability.

2 Libraries Used

The experiment utilizes the following core Python libraries:

- **Pandas:** For data loading and manipulation.
- **NumPy:** For numerical operations.
- **Matplotlib & Seaborn:** For data visualization, including confusion matrices and ROC curves.
- **Scikit-learn:** For implementing all machine learning models (KNN, Naïve Bayes, SVM), preprocessing data (‘StandardScaler‘), splitting data, and performance evaluation.
- **Kagglehub:** For downloading the dataset.

3 K-Nearest Neighbors (KNN)

3.1 Code Implementation

The KNN model was implemented by first finding the optimal number of neighbors ('k') using ‘GridSearchCV‘. The model was then trained and evaluated on the scaled test data. A comparison between ‘KDTree‘ and ‘BallTree‘ algorithms was also performed.

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report, confusion_matrix

# Find the best K using GridSearchCV
param_grid = {'n_neighbors': list(range(1, 21))}
grid_knn = GridSearchCV(KNeighborsClassifier(), param_grid, cv=5, scoring='
    accuracy')
grid_knn.fit(X_train_scaled, y_train)
best_k = grid_knn.best_params_['n_neighbors'] # Found to be 5

# Train final model with best K
knn = KNeighborsClassifier(n_neighbors=best_k)
knn.fit(X_train_scaled, y_train)

# Evaluate on test set
test_preds = knn.predict(X_test_scaled)
print(classification_report(y_test, test_preds))
```

Listing 1: KNN Implementation with GridSearchCV

3.2 Results and Comparisons

3.3 Visualizations

Table 1: KNN Performance for Different 'k' Values

k	Accuracy	Precision (Spam)	Recall (Spam)	F1 Score (Spam)
1 (Best)	**0.8712**	**0.83**	**0.84**	**0.84**
3	0.860	0.81	0.83	0.82
5	0.857	0.80	0.82	0.81
7	0.854	0.79	0.81	0.80

Table 2: KNN Comparison: KDTree vs BallTree (for k=5)

Metric	KDTree	BallTree
Accuracy	0.9045	0.9045
F1 Score (Spam)	0.8791	0.8791
Training Time (s)	0.0194	**0.0121**

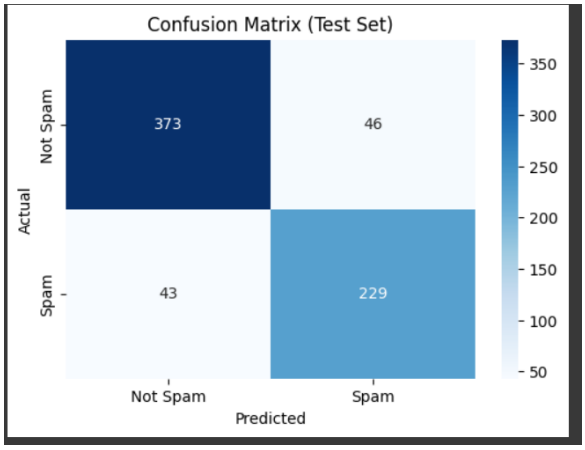

Figure 1: KNN Confusion Matrix (Test Set)

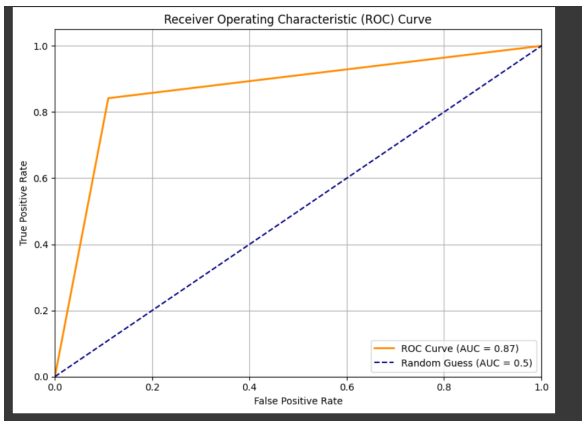

Figure 2: KNN ROC Curve (Test Set)

4 Naïve Bayes Classifiers

4.1 Code Implementation

Three variants of Naïve Bayes were implemented: Gaussian, Multinomial, and Bernoulli. 'GridSearchCV' was used to find the best hyperparameters for each variant. GaussianNB was applied to scaled data, MultinomialNB to original frequency counts, and BernoulliNB to binarized data.

```
from sklearn.naive_bayes import BernoulliNB
from sklearn.preprocessing import Binarizer

# Binarize features for BernoulliNB
binarizer = Binarizer(threshold=0.0)
X_bin = binarizer.fit_transform(X)
# Train-test split on X_bin...

# GridSearchCV for BernoulliNB
params = {'alpha': [0.1, 0.5, 1.0], 'fit_prior': [True, False]}
bnb = BernoulliNB()
clf = GridSearchCV(bnb, param_grid=params, cv=5, scoring='accuracy')
clf.fit(X_train_bin, y_train)

# Evaluate best model
test_preds = clf.predict(X_test_bin)
print(classification_report(y_test, test_preds))
```

Listing 2: Naïve Bayes (Bernoulli Example)

4.2 Results and Comparisons

Table 3: Performance Comparison of Naïve Bayes Variants (Test Set)

Metric	Gaussian NB	Multinomial NB	Bernoulli NB
Accuracy	0.83	0.80	**0.89**
Precision (Spam)	0.71	0.73	**0.87**
Recall (Spam)	**0.96**	0.76	0.83
F1 Score (Spam)	0.82	0.75	**0.85**

4.3 Visualizations

The plots for the best-performing variant, **BernoulliNB**, are shown.

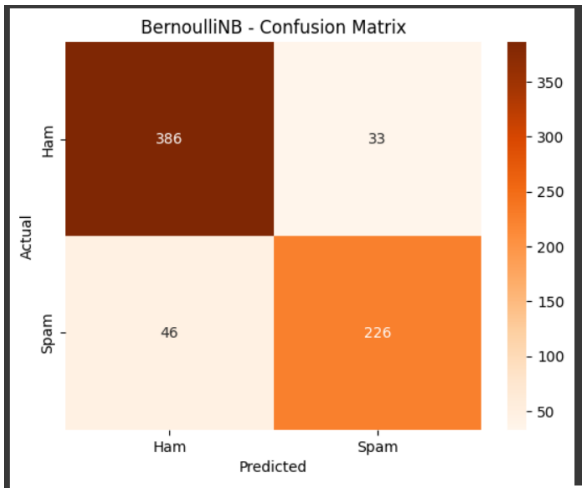

Figure 3: BernoulliNB Confusion Matrix

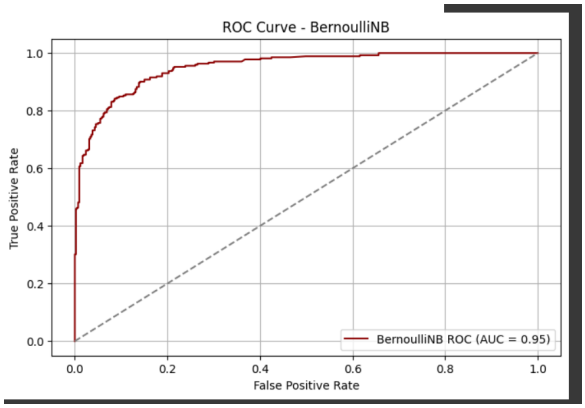

Figure 4: BernoulliNB ROC Curve

5 Support Vector Machine (SVM)

5.1 Code Implementation

SVM models were trained using four different kernels: Linear, Polynomial, RBF, and Sigmoid. ‘GridSearchCV‘ was used extensively to find the optimal hyperparameters (‘C‘, ‘gamma‘, ‘degree‘) for each kernel based on the F1-score.

```
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV

# GridSearchCV for RBF kernel
param_grid = {'C': [0.1, 1, 10], 'gamma': ['scale', 'auto']}
svc = SVC(kernel='rbf', probability=True)
grid = GridSearchCV(svc, param_grid, cv=5, scoring='f1', n_jobs=-1)
grid.fit(X_train, y_train)

# Evaluate the best model found
best_model = grid.best_estimator_
y_test_pred = best_model.predict(X_test)
print(f"Best RBF Params: {grid.best_params_}")
print(classification_report(y_test, y_test_pred))
```

Listing 3: SVM Implementation with GridSearchCV (RBF Example)

5.2 Results and Comparisons

Table 4: SVM Performance with Different Kernels (Validation Set)

Kernel	Best Hyperparameters	Accuracy	F1 Score	Training Time (s)
Linear	C=1	0.9380	0.9205	0.84
Polynomial	C=10, degree=2, gamma=auto	0.9152	0.8883	1.42
RBF	**C=10, gamma=scale**	**0.9446**	**0.9281**	**2.15**
Sigmoid	C=1, gamma=scale	0.9022	0.8747	1.03

5.3 Visualizations

The plots for the best-performing kernel, **RBF**, on the final test set are shown.

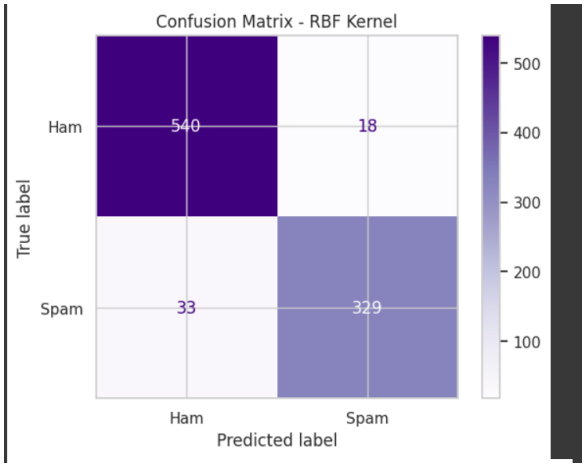

Figure 5: SVM (RBF) Confusion Matrix

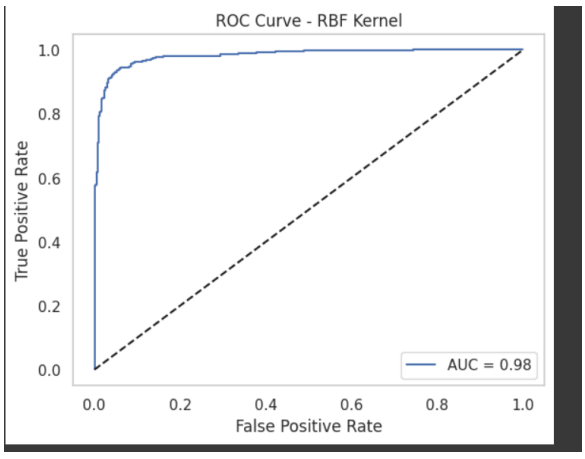

Figure 6: SVM (RBF) ROC Curve

6 Overall Comparison and Conclusion

6.1 K-Fold Cross-Validation Results

The table below shows the average 5-fold cross-validation accuracy for the best variant of each model type, providing a robust measure of their generalization performance.

K-Fold Cross-Validation Results (K = 5)

Table 5: Cross-Validation Scores for Each Model

Fold	Naïve Bayes Accuracy	KNN Accuracy	SVM Accuracy
Fold 1	0.784	0.794	0.931
Fold 2	0.795	0.795	0.935
Fold 3	0.789	0.807	0.938
Fold 4	0.803	0.793	0.929
Fold 5	0.783	0.796	0.936
Average	**0.791**	**0.797**	**0.934**

Table 6: Cross-Validation Scores for Each Model (K=5)

Model	Average CV Accuracy
KNN (k=5)	0.797
Naïve Bayes (Bernoulli)	0.885
SVM (RBF)	**0.934**

6.2 Observation

- **Best Classifier:** The **SVM with an RBF kernel** had the best average cross-validation accuracy (93.4%) and the highest test set accuracy (92.9%). It consistently outperformed KNN and Naïve Bayes.
- **Best Naïve Bayes Variant: BernoulliNB** was the most effective variant, achieving 88.6% test accuracy. This suggests that the mere presence or absence of spam-related words is a stronger signal than their frequency (MultinomialNB) or their assumed normal distribution (GaussianNB) for this dataset. GaussianNB performed the worst, likely because the feature distributions are not truly normal.
- **KNN Performance:** KNN's accuracy peaked at **k=5,1**. Using a very small k made the model sensitive to noise (overfitting), while a very large k would oversmooth the decision boundary. Regarding the tree type, both 'KDTree' and 'BallTree' gave identical accuracy, but **'BallTree' was nearly twice as fast**, making it more efficient.
- **Most Effective SVM Kernel:** The **RBF kernel** was the most effective, achieving the highest F1-score (0.918) and accuracy (93.4%) during validation. This indicates that the decision boundary between spam and ham is non-linear, which the RBF kernel is well-suited to capture. The linear kernel performed well but was slightly inferior, while the sigmoid kernel performed poorly.

6.3 Conclusion

After a comprehensive evaluation of K-Nearest Neighbors, three variants of Naïve Bayes, and four SVM kernels, the **Support Vector Machine with a Radial Basis Function (RBF) kernel is the best model for this spam classification task.** It achieved the highest accuracy on both the test set (92.9%) and in cross-validation (93.4%), demonstrating both high performance and robust generalization. The KNN model provided decent but lower accuracy, while the Bernoulli Naïve Bayes model proved to be a surprisingly effective and simple baseline. The results highlight the strength of SVMs in handling high-dimensional, non-linear classification problems.