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)	Batch: 2023-2028	**Due date:**

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

1 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 and K-Fold cross-validation.

2 Libraries used:

- Pandas
- Numpy
- Matplotlib
- Scikit-learn
- Time

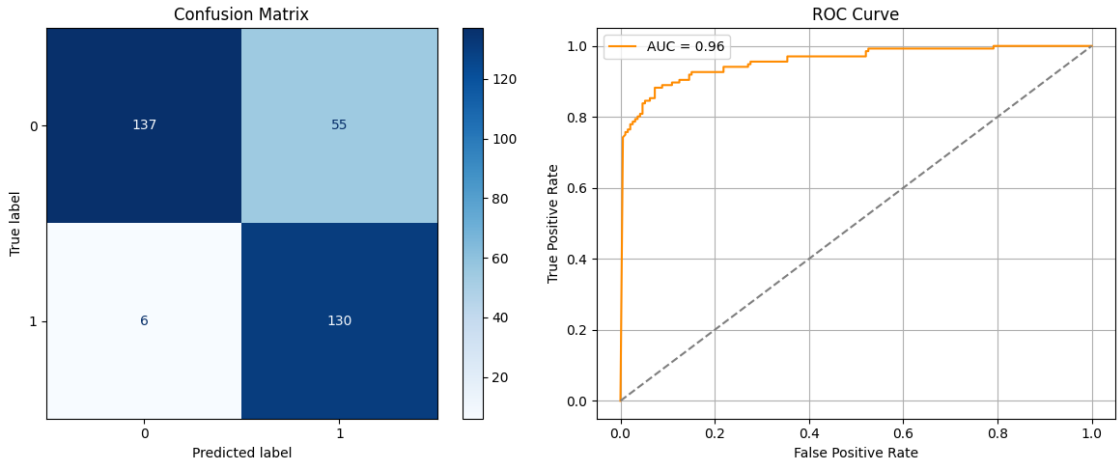
3 Objective:

The objective of this assignment is to implement and compare the performance of three classification algorithms—Naïve Bayes, K-Nearest Neighbors (KNN), and Support Vector Machine (SVM)—for classifying emails as spam or ham. This includes preprocessing the dataset, training the models using K-Fold cross-validation, and evaluating them based on accuracy metrics.

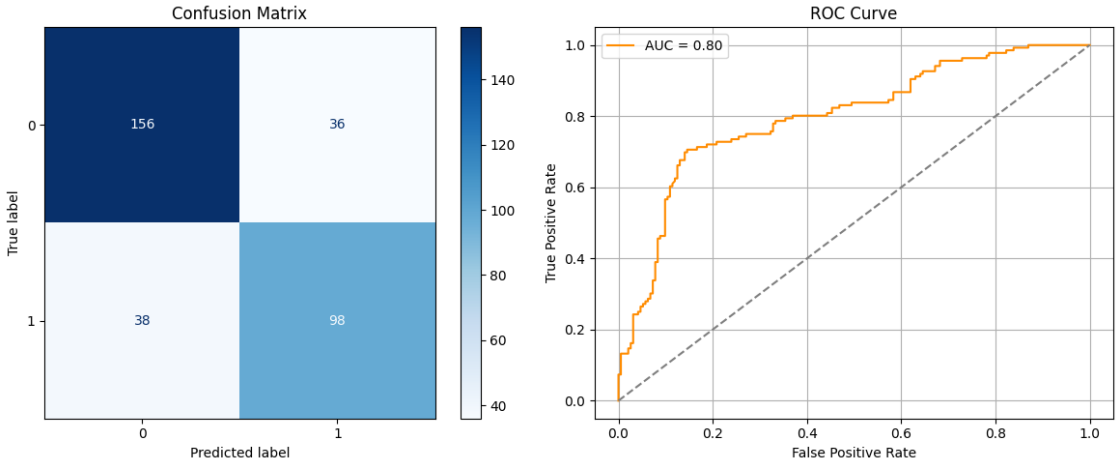
4 Code with Plot

```
# TRAINING + EVALUATION CODE OF GIVEN MODEL
def evaluate_model(name, model, X_train, X_test):
    start_time = time.time()
    model.fit(X_train, y_train)
```

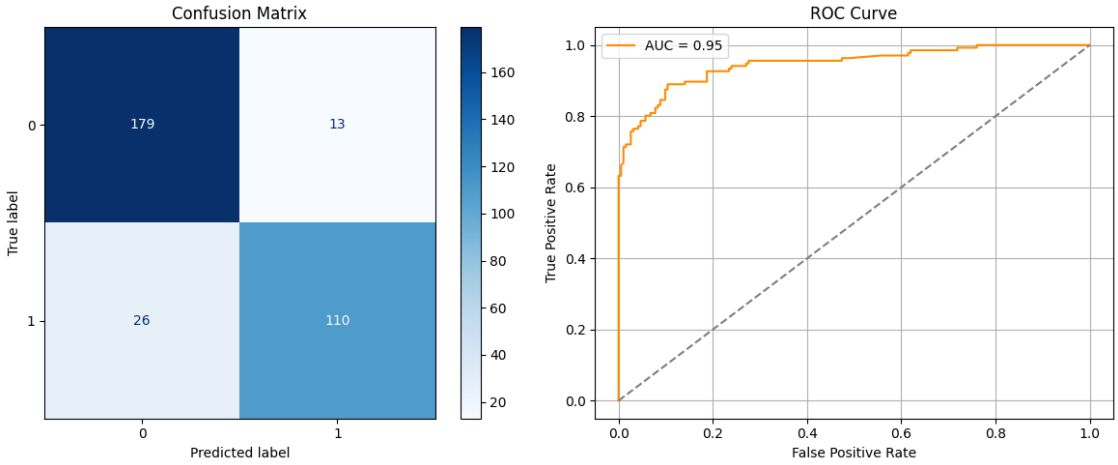
```
    end_time = time.time()
    y_pred = model.predict(X_test)
    print(f"\n{name} Performance:")
    print(f"Accuracy :  {accuracy_score(y_test, y_pred):.4f}")
    print(f"Precision:  {precision_score(y_test, y_pred, average='macro'):.4f}")
    print(f"Recall   :  {recall_score(y_test, y_pred, average='macro'):.4f}")
    print(f"F1 Score :  {f1_score(y_test, y_pred, average='macro'):.4f}")
    print(f"Training Time: {(end_time - start_time):.4f} seconds")

    fig, axes = plt.subplots(1, 2, figsize=(12, 5))

    cm = confusion_matrix(y_test, y_pred)
    disp = ConfusionMatrixDisplay(confusion_matrix=cm)
    disp.plot(ax=axes[0], cmap='Blues')
    axes[0].set_title('Confusion Matrix')

    y_prob = model.predict_proba(X_test)[:, 1]
    fpr, tpr, _ = roc_curve(y_test, y_prob)
    roc_auc = auc(fpr, tpr)

    axes[1].plot(fpr, tpr, label=f'AUC = {roc_auc:.2f}', color='darkorange')
    axes[1].plot([0, 1], [0, 1], linestyle='--', color='gray')
    axes[1].set_xlabel('False Positive Rate')
    axes[1].set_ylabel('True Positive Rate')
    axes[1].set_title('ROC Curve')
    axes[1].legend()
    axes[1].grid(True)
    plt.tight_layout()
    plt.show()


# NAIVE BAYES - GAUSSIAN
evaluate_model("GaussianNB", GaussianNB(), X_train, X_test)
```



```
# NAIVE BAYES - MULTINOMIAL
evaluate_model("MultinomialNB", MultinomialNB(), X_train, X_test)
```




```
# NAIVE BAYES - BERNOULLI
evaluate_model("BernoulliNB", BernoulliNB(), X_train, X_test)
```




```
# K-NEAREST NEIGHBOURS - VARYING K VALUES[1, 3, 5, 7]
for k in [1, 3, 5, 7]:
    evaluate_model(f"KNN (k={k})", KNeighborsClassifier(n_neighbors=k), X_train, X_test)
```

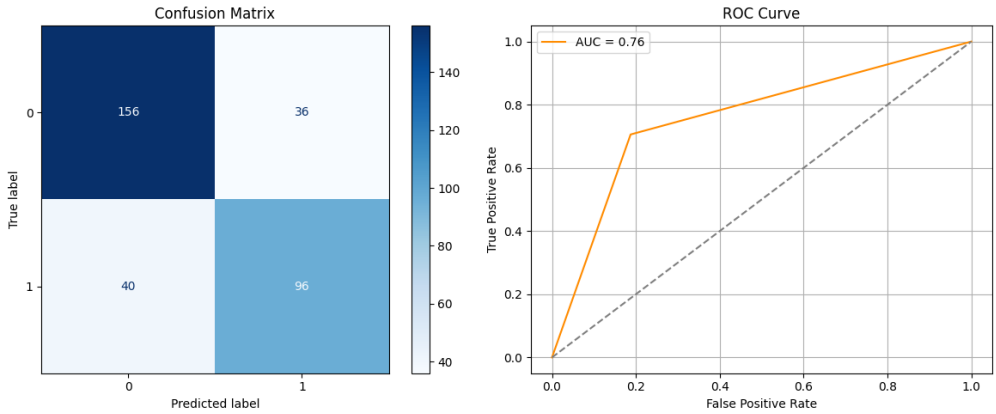


Figure 1: k = 1

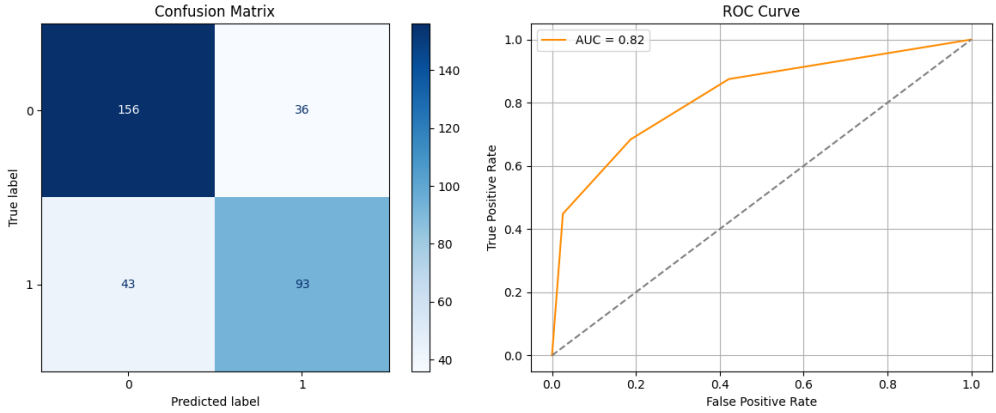


Figure 2: k = 3

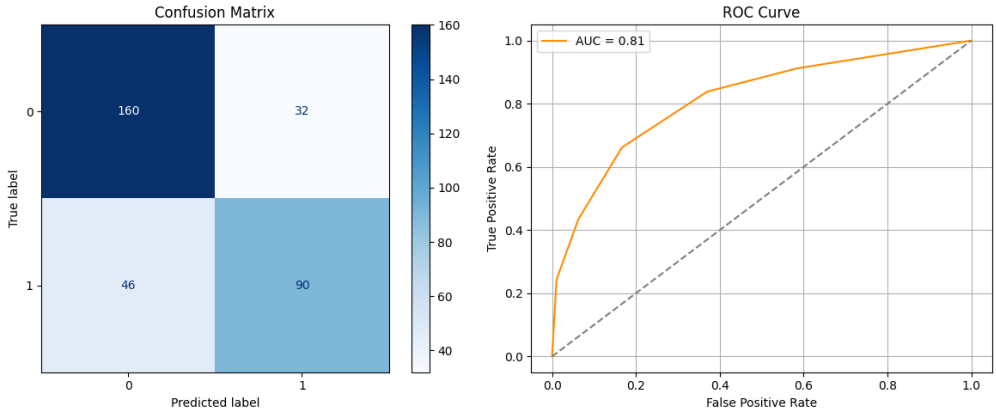


Figure 3: k = 5

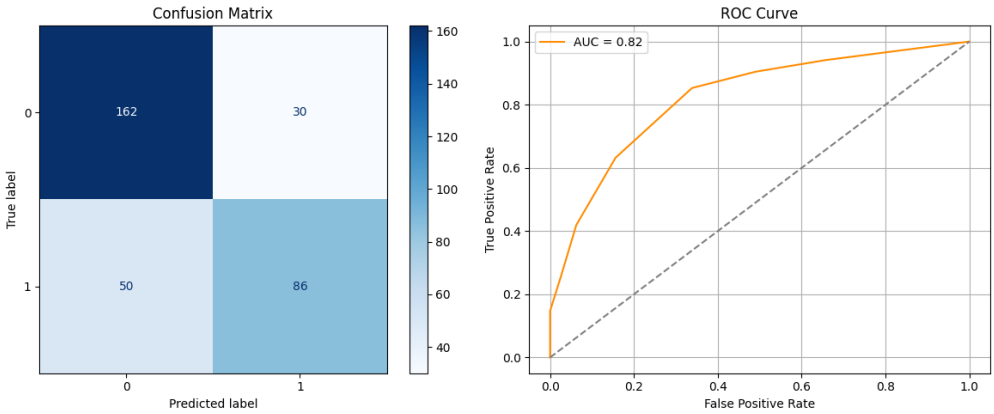
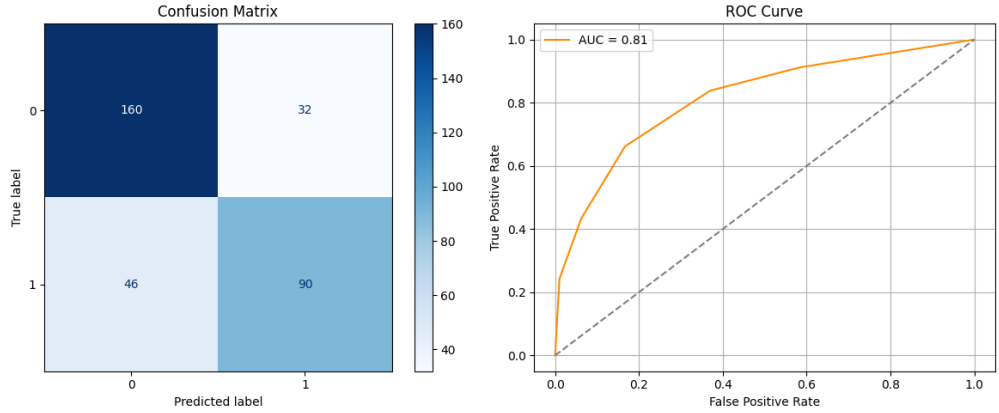
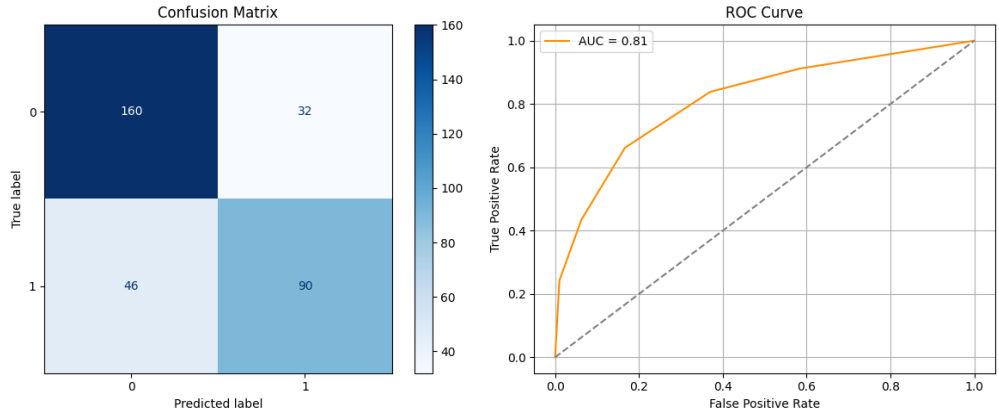


Figure 4: k = 7

```
# K-NEAREST NEIGHBOURS - KDTREE
evaluate_model("KNN (KDTree)", KNeighborsClassifier(algorithm='kd_tree'), X_train, X_test)
```




```
# K-NEAREST NEIGHBOURS - BALLTREE
evaluate_model("KNN (BallTree)", KNeighborsClassifier(algorithm='ball_tree'), X_train, X_test)
```



5 Comparison Tables:

Naïve Bayes Variant Comparison:

Metric	Gaussian NB	Multinomial NB	Bernoulli NB
Accuracy	0.8140	0.7744	0.8811
Precision	0.8304	0.7677	0.8837
Recall	0.8347	0.7665	0.8706
F1 Score	0.8139	0.7671	0.8756

Table 1: Performance Comparison of Naïve Bayes Variants

KNN: Varying k Values

k	Accuracy	Precision	Recall	F1 Score
1	0.7683	0.7616	0.7592	0.7603
3	0.7591	0.7524	0.7482	0.7499
5	0.7622	0.7572	0.7475	0.7508
7	0.7561	0.7528	0.7381	0.7423

Table 2: KNN Performance for Different k Values

KNN: KDTree vs BallTree

Metric	KDTree	BallTree
Accuracy	0.7622	0.7622
Precision	0.7572	0.7572
Recall	0.7475	0.7475
F1 Score	0.7508	0.7508
Training Time(s)	0.0117	0.0089

Table 3: KNN Comparison : KDTree vs BallTree

SVM Kernel-wise Results

Kernel	Hyperparameters	Accuracy	F1 Score	Training Time
Linear				
Polynomial				
RBF				
Sigmoid				

Table 4: SVM Performance with Different Kernels and Parameters

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

Fold	Naïve Bayes Accuracy	KNN Accuracy	SVM Accuracy
Fold 1			
Fold 2			
Fold 3			
Fold 4			
Fold 5			
Average			

Table 5: Cross-Validation Scores for Each Model

6 Observations and Conclusions

- **Which classifier had the best average accuracy?**
- **Which Naïve Bayes variant worked best?**
- **How did KNN accuracy vary with k and tree type?**
- **Which SVM kernel was most effective?**
- **How did hyperparameters influence performance?**

7 Learning Outcomes

- How to implement Naïve Bayes, K-Nearest Neighbors (KNN), and Support Vector Machine (SVM) for binary classification.
- How to evaluate classifiers using accuracy and other performance metrics like precision, recall, and F1-score.
- How to compare multiple models and select the best one for a spam detection task.

GitHub Repository:
https://github.com/vidarshanaa15/ml-expt-3