Experiment 2: Spam or Ham Classification using Naïve Bayes, KNN, and SVM

Machine Learning Lab Report

Academic Year 2025–2026

Aim

To classify emails as spam or ham using three classification algorithms: Naïve Bayes, K-Nearest Neighbors (KNN), and Support Vector Machine (SVM), and to evaluate their performance using standard classification metrics and K-Fold cross-validation.

Libraries Used

- pandas
- numpy
- matplotlib
- seaborn
- scikit-learn

Objective

- Load and preprocess the dataset
- Perform exploratory data analysis (EDA)
- Train classifiers: Naïve Bayes (Gaussian, Multinomial, Bernoulli), KNN, and SVM
- Evaluate using accuracy, precision, recall, F1-score, confusion matrix, and ROC curve
- Compare performance using 5-fold cross-validation

Code Snippets

1. Data Loading and Preprocessing:

```
df = pd.read_csv("spambase_csv.csv")
df.fillna(df.mean(), inplace=True)
X_raw = df.drop(columns=['class'])
y = df['class']
```

2. Train-Test Split and Scaling:

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

X_train_raw, X_test_raw, y_train_raw, y_test_raw = train_test_split(
    X_raw, y, test_size=0.2, random_state=42
)

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_raw)
X_train_scaled, X_test_scaled, y_train_scaled, y_test_scaled =
   train_test_split(
    X_scaled, y, test_size=0.2, random_state=42
)
```

3. Evaluation Function:

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

def evaluate(name, model, X_test, y_test):
    y_pred = model.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    prec = precision_score(y_test, y_pred)
    rec = recall_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)
    print(f"{name} -- Accuracy: {acc:.2f}, Precision: {prec:.2f}, Recall:
   {rec:.2f}, F1 Score: {f1:.2f}")
```

4. Training Gaussian Naïve Bayes:

```
from sklearn.naive_bayes import GaussianNB
model = GaussianNB()
model.fit(X_train_raw, y_train_raw)
evaluate("GaussianNB", model, X_test_raw, y_test_raw)
```

5. KNN with Different k Values:

```
from sklearn.neighbors import KNeighborsClassifier
for k in [1, 3, 5, 7]:
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train_scaled, y_train_scaled)
    evaluate(f"KNN (k={k})", knn, X_test_scaled, y_test_scaled)
```

6. SVM with Different Kernels:

```
from sklearn.svm import SVC
model = SVC(kernel='rbf', probability=True)
model.fit(X_train_scaled, y_train_scaled)
evaluate("SVM - RBF", model, X_test_scaled, y_test_scaled)
```

Screenshots of Outputs

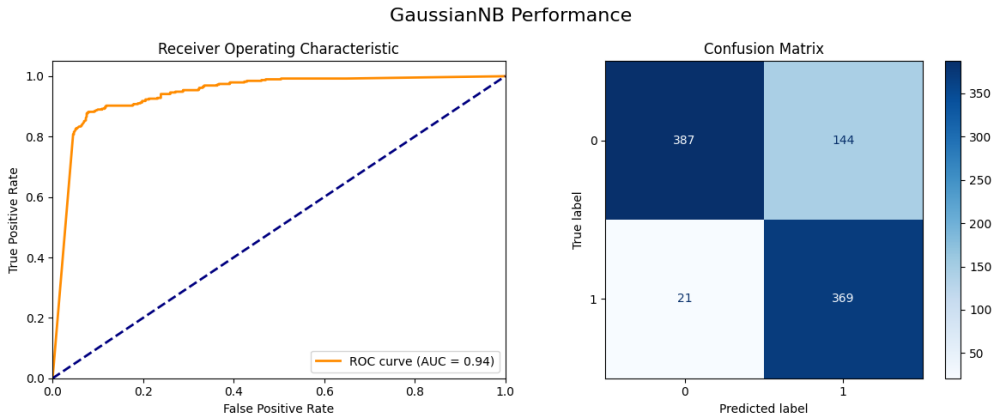

Figure 1: ROC Curve and Confusion Matrix - Naïve Bayes GaussianNB

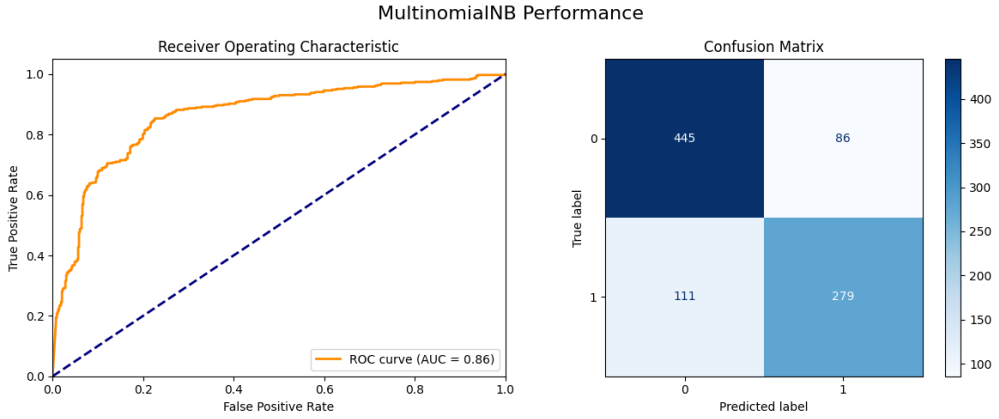

Figure 2: ROC Curve and Confusion Matrix - Naïve Bayes MultinomialNB

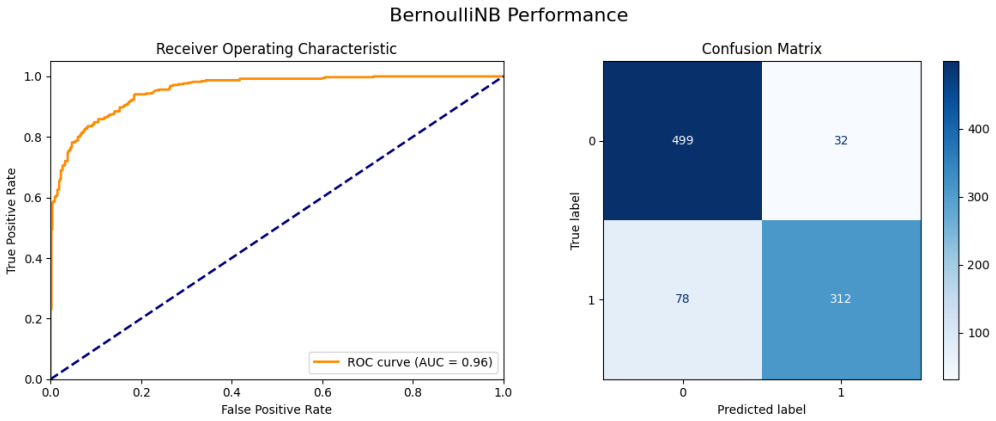

Figure 3: ROC Curve and Confusion Matrix - Naïve Bayes BernoulliNB

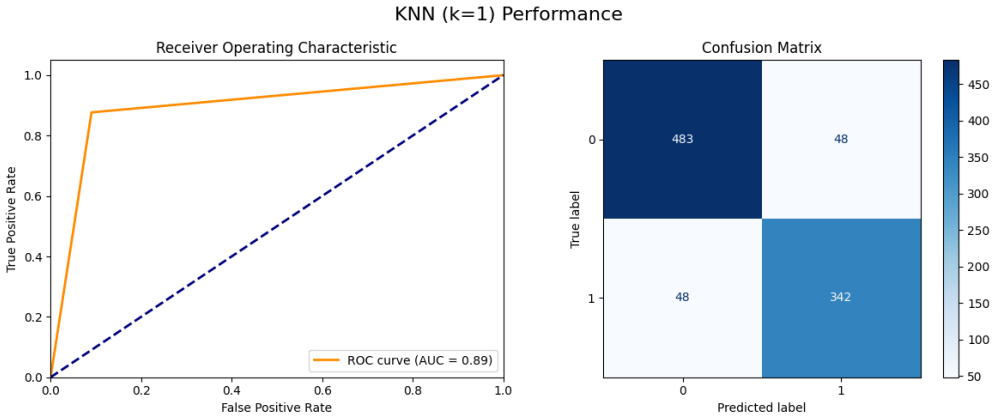

Figure 4: ROC Curve and Confusion Matrix - KNN (K=1)

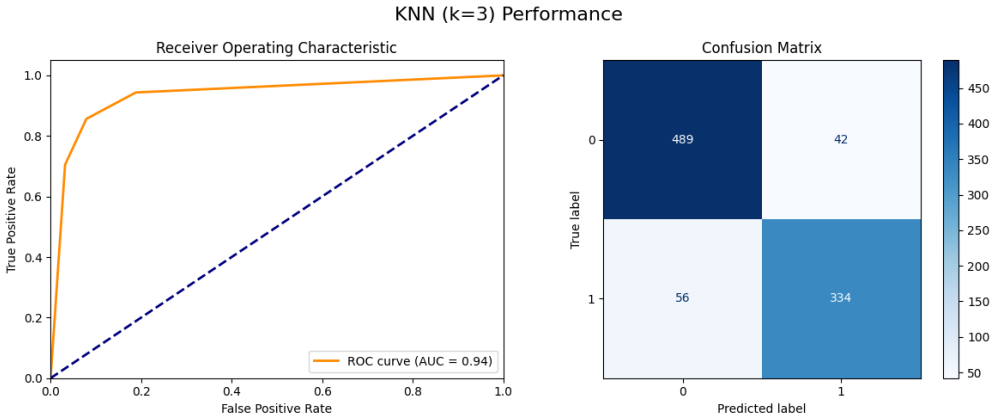

Figure 5: ROC Curve and Confusion Matrix - KNN (K=3)

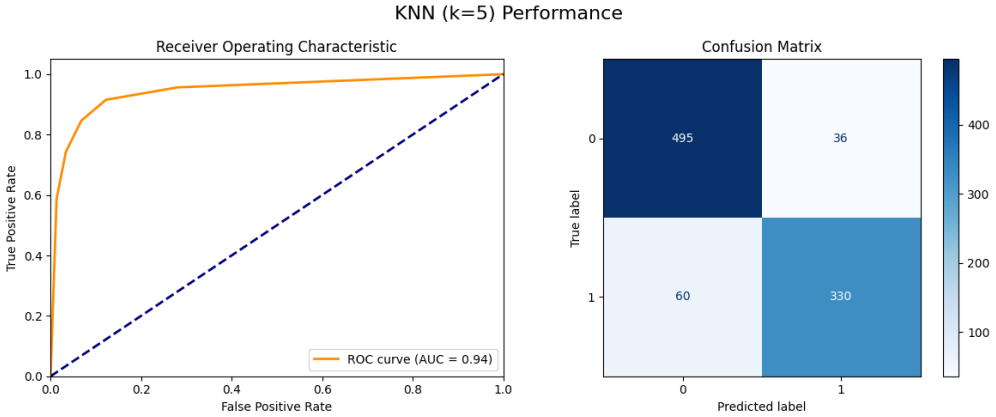

Figure 6: ROC Curve and Confusion Matrix - KNN (K=5)

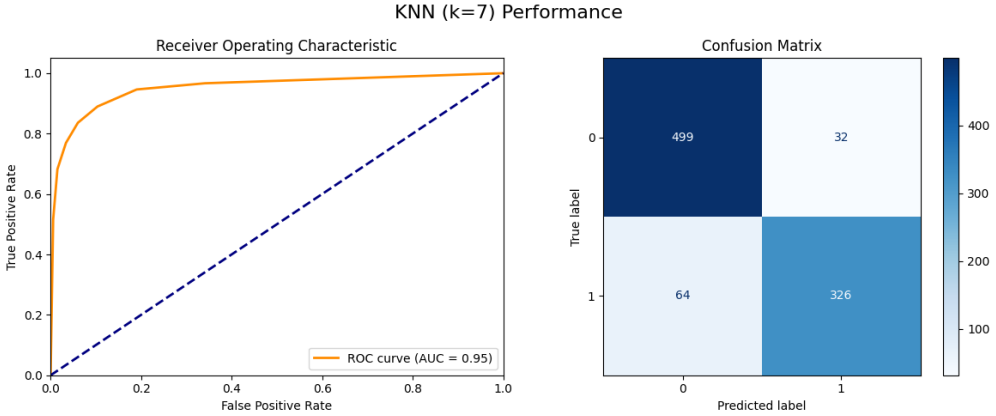

Figure 7: ROC Curve and Confusion Matrix - KNN (K=7)

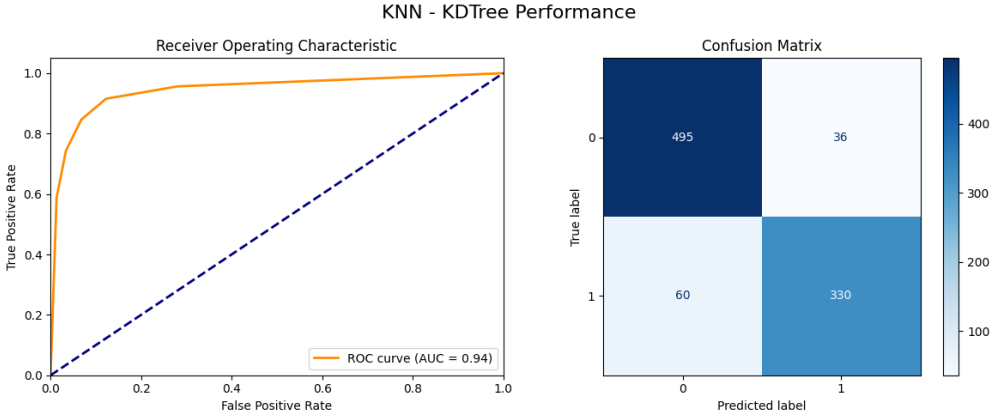

Figure 8: ROC Curve and Confusion Matrix - KNN (KDTree)

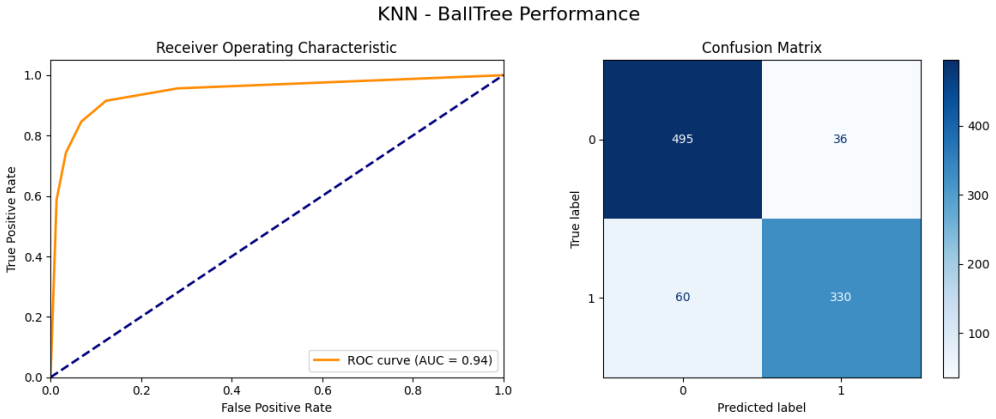

Figure 9: ROC Curve and Confusion Matrix - KNN (BallTree)

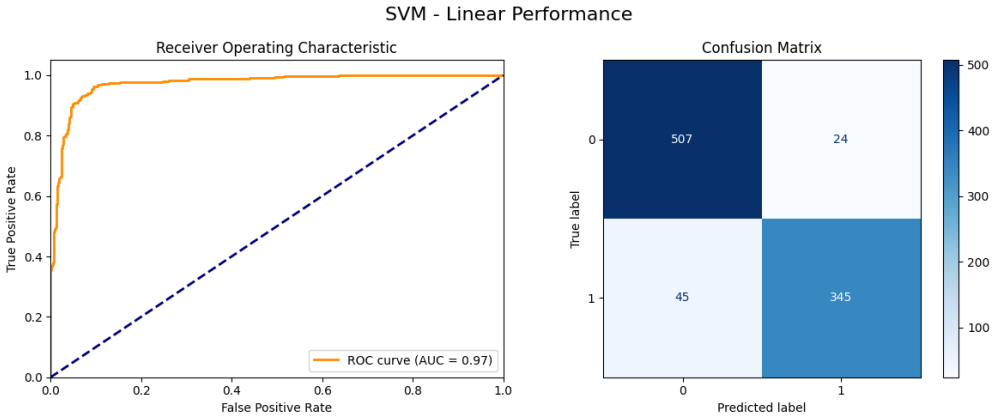

Figure 10: ROC Curve and Confusion Matrix - SVM Linear

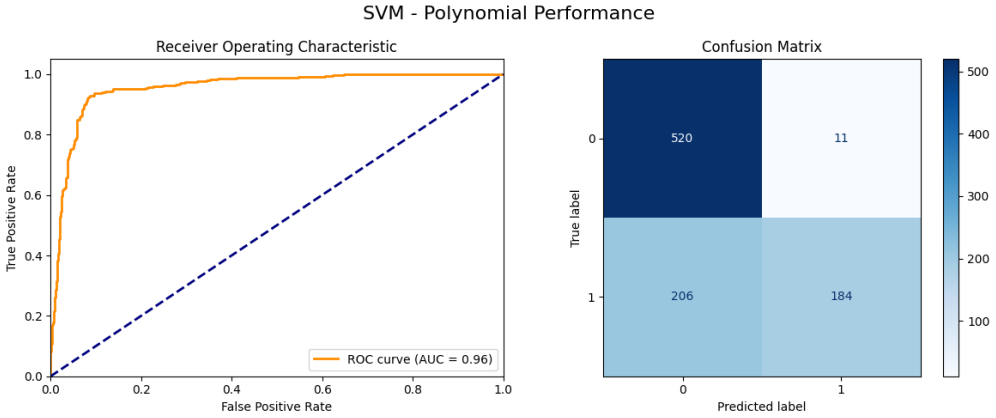

Figure 11: ROC Curve and Confusion Matrix - SVM Polynomial

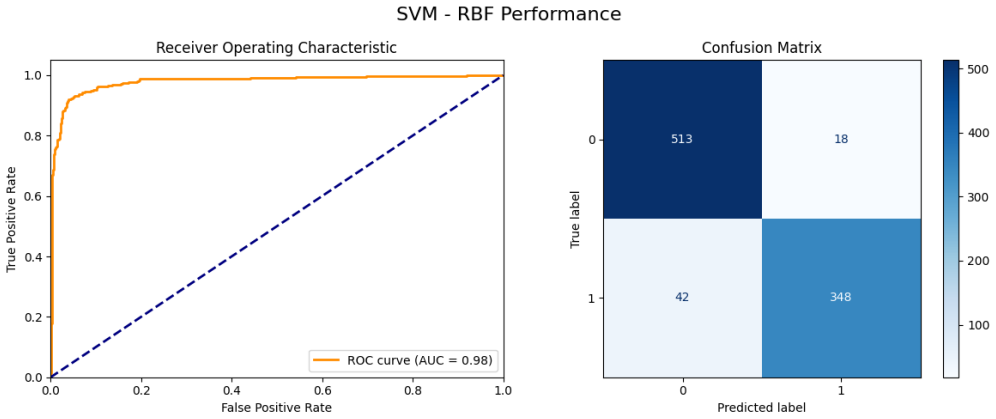

Figure 12: ROC Curve and Confusion Matrix - SVM RBF

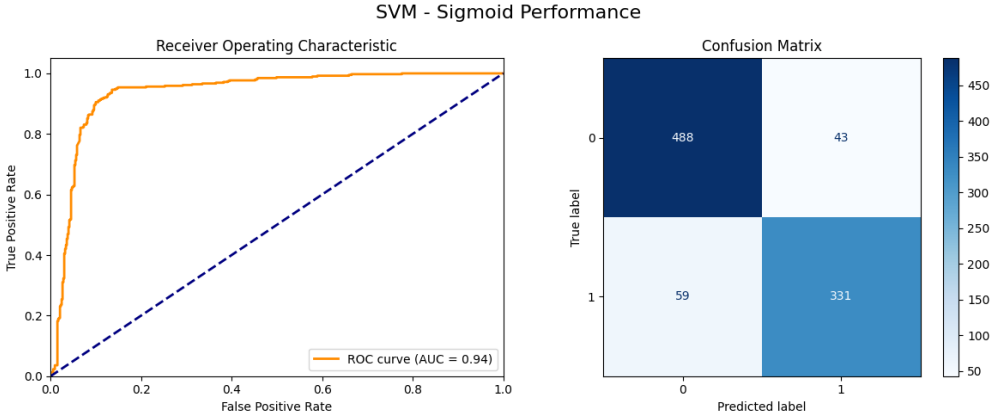

Figure 13: ROC Curve and Confusion Matrix - SVM Sigmoid

Results and Comparisons

Table 1: Naïve Bayes Variant Comparison

Metric	Gaussian NB	Multinomial NB	Bernoulli NB
Accuracy	0.820847	0.786102	0.880565
Precision	0.719298	0.764384	0.906977
Recall	0.946154	0.715385	0.800000
F1 Score	0.817276	0.739073	0.850136

Table 2: KNN Performance for Different k

k	Accuracy	Precision	Recall	F1 Score
1	0.895765	0.876923	0.876923	0.876923
3	0.893594	0.888298	0.856140	0.872063
5	0.895765	0.901639	0.846154	0.873016
7	0.895765	0.910615	0.835897	0.871658

Table 3: KNN Comparison: KDTree vs BallTree

Metric	KDTree	BallTree
Accuracy	0.895765	0.895765
Precision	0.901639	0.901639
Recall	0.846154	0.846154
F1 Score	0.873016	0.873016
Training Time (s)	0.812832	0.808676

Table 4: SVM Performance with Different Kernels

Kernel	Hyperparameters	Accuracy	F1 Score	Train Time (s)
Linear	C=1	0.925081	0.909091	2.467797
Polynomial	C=1, degree=3, gamma=scale	0.764387	0.629060	2.901383
RBF	C=1, gamma=scale	0.934853	0.920635	2.408404
Sigmoid	C=1, gamma=scale	0.889251	0.866492	1.829148

Table 5: K-Fold Cross-Validation Accuracy (K=5)

Fold	**Naïve Bayes**	**KNN (k=5)**	**SVM (RBF)**
Fold 1	0.820847	0.895765	0.934853
Fold 2	0.817391	0.904348	0.933696
Fold 3	0.801087	0.929348	0.922826
Fold 4	0.820652	0.903261	0.935870
Fold 5	0.835870	0.909783	0.930435
Average	0.819169	0.908501	0.931536

Conclusion

In this experiment, we successfully implemented and evaluated three major classification algorithms: Naïve Bayes, K-Nearest Neighbors (KNN), and Support Vector Machine (SVM) on the Spambase dataset. Naïve Bayes was the fastest and performed well with Gaussian and Multinomial variants. KNN showed high accuracy with lower values of k, but performance degraded as k increased. SVM, particularly with the RBF kernel, achieved the highest accuracy and F1-score overall. Standardization significantly improved KNN and SVM performance. ROC curves and confusion matrices highlighted SVM's strong classification boundary. Thus, SVM with RBF kernel is most suitable for this spam classification task.