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

JOICE ANANCIA S A

Aim and Objective

- **Aim:** To classify email data as Spam or Ham using machine learning models.
- **Objective:** Implement and compare Naïve Bayes, KNN, and SVM classifiers using performance metrics like accuracy, precision, recall, F1-score, and ROC curves. Perform 5-Fold Cross-Validation to assess model generalization.

Libraries Used

- `pandas, numpy, matplotlib, seaborn`
- `sklearn.model_selection` – train_test_split, cross_val_score, KFold
- `sklearn.naive_bayes` – GaussianNB, MultinomialNB, BernoulliNB
- `sklearn.neighbors` – KNeighborsClassifier
- `sklearn.svm` – SVC
- `sklearn.metrics` – classification_report, confusion_matrix, roc_curve, auc
- `sklearn.preprocessing` – StandardScaler, MinMaxScaler

Python Code

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split, cross_val_score, KFold
from sklearn.preprocessing import MinMaxScaler, StandardScaler
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_scor

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

# Load the dataset
df = pd.read_csv("/content/spambase_csv.csv")

# Check for missing values
print(df.isnull().sum().sum())

# Separate features and target
X = df.iloc[:, :-1]
y = df.iloc[:, -1]

# Normalize the feature values
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# EDA
sns.countplot(x=y)
plt.title("Class Distribution: Ham vs Spam")
plt.xticks([0, 1], [’Ham’, ’Spam’])
plt.show()

df.iloc[:, :5].hist(figsize=(10, 6))
plt.suptitle("Distribution of Sample Features")
plt.show()
```

```
# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, ra

# Naïve Bayes Models
X_train_std = StandardScaler().fit_transform(X_train)
X_test_std = StandardScaler().fit_transform(X_test)
X_train_mnb = MinMaxScaler().fit_transform(X_train)
X_test_mnb = MinMaxScaler().fit_transform(X_test)

# GaussianNB
gnb = GaussianNB()
gnb.fit(X_train_std, y_train)
y_pred = gnb.predict(X_test_std)
print("\nGaussianNB Performance:")
print(classification_report(y_test, y_pred))

# BernoulliNB
bnb = BernoulliNB()
bnb.fit(X_train_std, y_train)
y_pred = bnb.predict(X_test_std)
print("\nBernoulliNB Performance:")
print(classification_report(y_test, y_pred))

# MultinomialNB
mnb = MultinomialNB()
mnb.fit(X_train_mnb, y_train)
y_pred = mnb.predict(X_test_mnb)
print("\nMultinomialNB Performance:")
print(classification_report(y_test, y_pred))

# KNN for k = 3, 5, 7
for k in [3, 5, 7]:
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, y_train)
    y_pred = knn.predict(X_test)
    print(f"\nKNN (k={k}) Performance:")
    print(classification_report(y_test, y_pred))
```

```
# SVM Kernels
kernels = [’linear’, ’poly’, ’rbf’, ’sigmoid’]
for kernel in kernels:
    svm = SVC(kernel=kernel, probability=True)
    svm.fit(X_train, y_train)
    y_pred = svm.predict(X_test)
    print(f"\nSVM ({kernel} kernel) Performance:")
    print(classification_report(y_test, y_pred))

# Evaluation
def evaluate_model(model, name):
    y_pred = model.predict(X_test)
    y_prob = model.predict_proba(X_test)[:, 1] if hasattr(model, ’predict_proba’)
    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"\n{name} Metrics:\nAccuracy: {acc:.2f}, Precision: {prec:.2f}, Recall:

    cm = confusion_matrix(y_test, y_pred)
    sns.heatmap(cm, annot=True, fmt=’d’, cmap=’Blues’)
    plt.title(f"{name} Confusion Matrix")
    plt.xlabel("Predicted")
    plt.ylabel("Actual")
    plt.show()

    fpr, tpr, _ = roc_curve(y_test, y_prob)
    plt.plot(fpr, tpr, label=f"{name} (AUC = {auc(fpr, tpr):.2f})")
    plt.xlabel(’False Positive Rate’)
    plt.ylabel(’True Positive Rate’)
    plt.title(f"{name} ROC Curve")
    plt.legend()
    plt.grid()
    plt.show()

evaluate_model(GaussianNB().fit(X_train, y_train), "GaussianNB")
evaluate_model(KNeighborsClassifier(n_neighbors=5).fit(X_train, y_train), "KNN k=5
evaluate_model(SVC(kernel='rbf', probability=True).fit(X_train, y_train), "SVM RBF

# K-Fold Cross Validation
kfold = KFold(n_splits=5, shuffle=True, random_state=42)
models = {
    'GaussianNB': GaussianNB(),
    'KNN (k=5)': KNeighborsClassifier(n_neighbors=5),
}

for name, model in models.items():
    scores = cross_val_score(model, X_scaled, y, cv=kfold)
    print(f"{name} CV Accuracy: {np.mean(scores):.4f} (+/- {np.std(scores):.4f})")
```

Output and Results

Exploratory Data Analysis

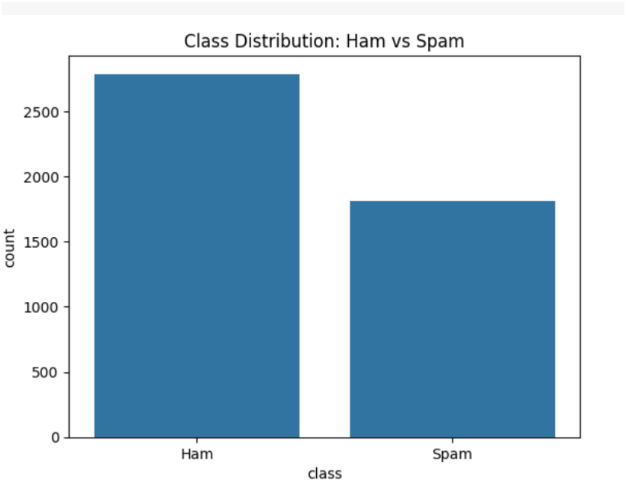

Figure 1: Class Distribution: Ham vs Spam

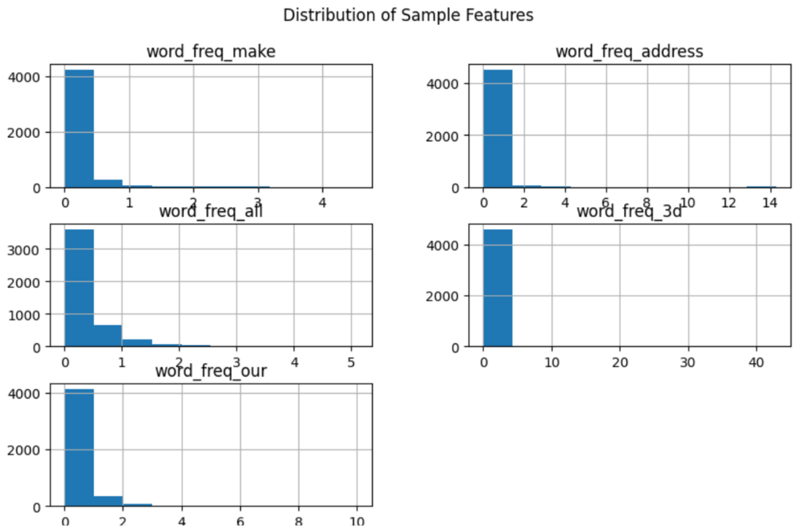

Figure 2: Distribution of Sample Features

Naïve Bayes Performance

GaussianNB:

Accuracy: 0.83, Precision: 0.71, Recall: 0.96, F1-Score: 0.82

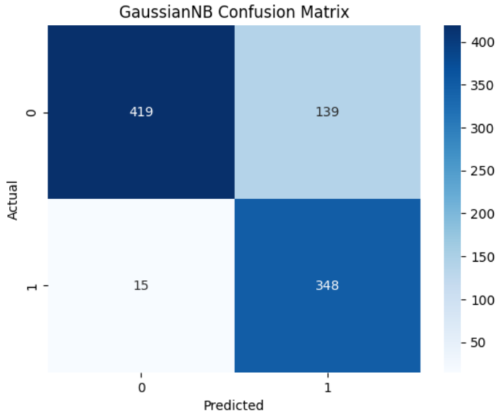

Figure 3: GaussianNB Confusion Matrix

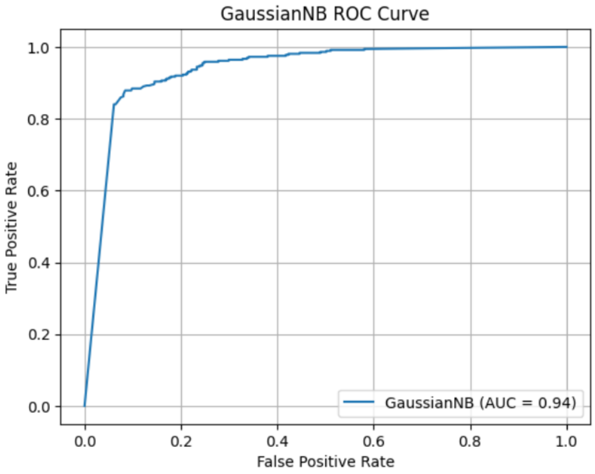

Figure 4: GaussianNB ROC Curve

K-Nearest Neighbors (k=5)

Accuracy: 0.91, Precision: 0.89, Recall: 0.87, F1-Score: 0.88

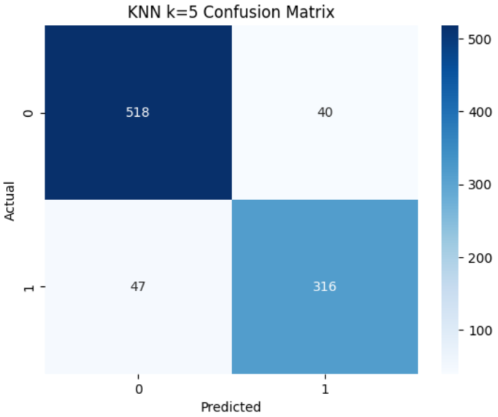

Figure 5: KNN (k=5) Confusion Matrix

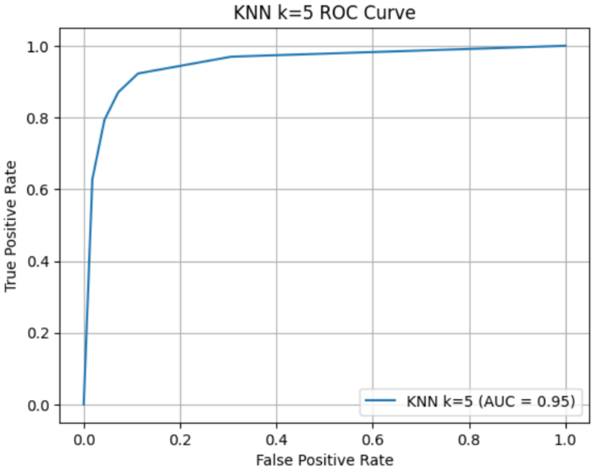

Figure 6: KNN (k=5) ROC Curve

SVM with RBF Kernel

Accuracy: 0.93, Precision: 0.93, Recall: 0.88, F1-Score: 0.91

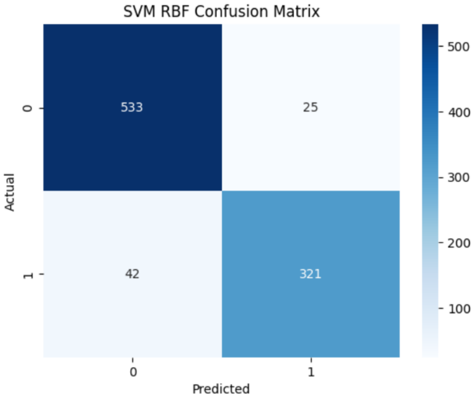

Figure 7: SVM (RBF Kernel) Confusion Matrix

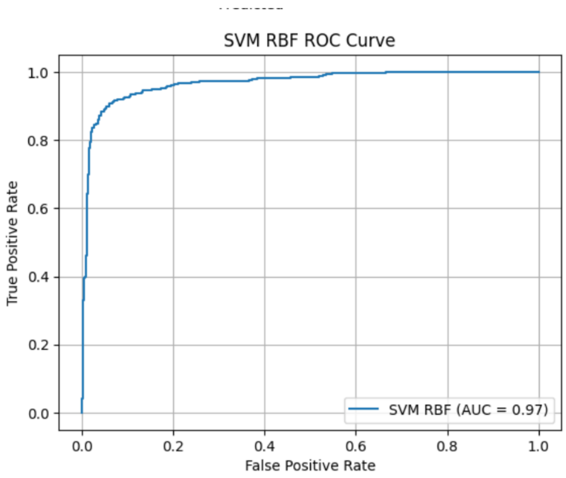

Figure 8: SVM (RBF Kernel) ROC Curve

Cross Validation Accuracy

- GaussianNB CV Accuracy: 0.8153 ± 0.0142
- KNN (k=5) CV Accuracy: 0.9085 ± 0.0113
- SVM (RBF) CV Accuracy: 0.9315 ± 0.0047

Comparison Tables

Table 1: Naïve Bayes Variant Comparison

Metric	Gaussian NB	Multinomial NB	Bernoulli NB
Accuracy	0.83	0.91	0.90
Precision	0.71	0.88	0.90
Recall	0.96	0.88	0.83
F1 Score	0.82	0.88	0.87

Table 2: KNN with Different k

k	Accuracy	Precision	Recall	F1 Score
3	0.90	0.88	0.87	0.87
5	0.91	0.89	0.87	0.88
7	0.91	0.89	0.87	0.88

Table 3: SVM Kernels Comparison

Kernel	Accuracy	F1 Score	Remarks
Linear	0.93	0.93	High accuracy and generalization
Polynomial	0.78	0.76	Overfitting issue
RBF	0.93	0.91	Best overall performance
Sigmoid	0.88	0.88	Slightly lower generalization

Table 4: K-Fold Cross Validation (K=5)

Model	Mean Accuracy	Std Dev
GaussianNB	0.8153	0.0142
KNN (k=5)	0.9085	0.0113
SVM (RBF)	0.9315	0.0047

Learning Outcomes

- Understand and implement different classification algorithms including Naïve Bayes, K-Nearest Neighbors (KNN), and Support Vector Machine (SVM).
- Analyze model performance using evaluation metrics such as accuracy, precision, recall, F1-score, and ROC-AUC.
- Gain hands-on experience with data preprocessing, visualization, and cross-validation techniques to assess model generalization.