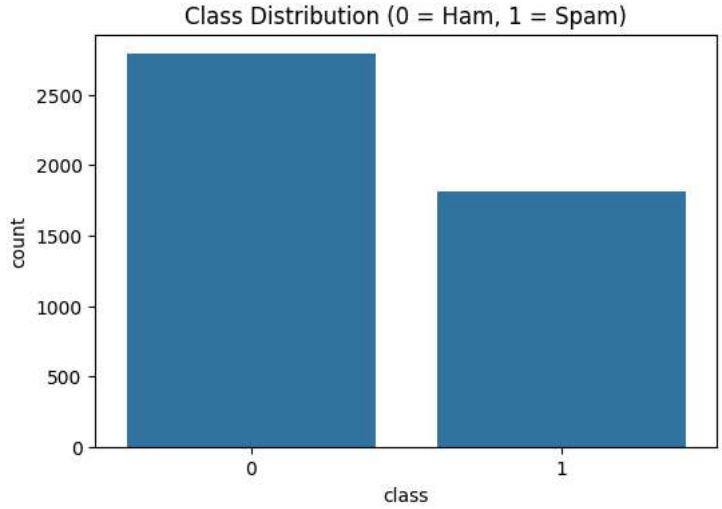
```
from google.colab import files
uploaded = files.upload()
```

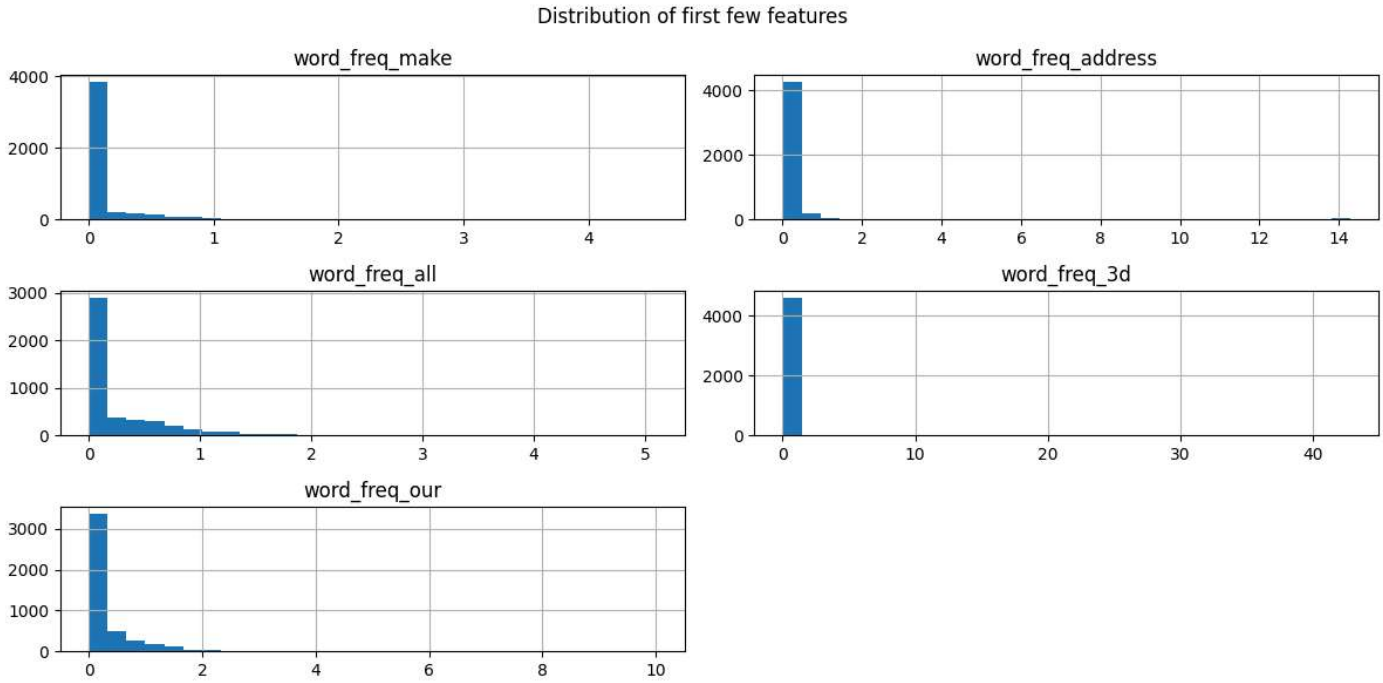
```
Choose Files spambase_csv.csv
• spambase_csv.csv(text/csv) - 703870 bytes, last modified: 8/1/2025 - 100% done
Saving spambase_csv.csv to spambase_csv (3).csv
```

```
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.naive_bayes import GaussianNB
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix, classification_report, roc_curve, auc



print("Gaussian (naive bayes)")
# 1. Load Dataset
df = pd.read_csv('spambase_csv.csv')
print("Columns in dataset:\n", df.columns.tolist())
print("Dataset shape:", df.shape)

# 2. Separate features and target
# The last column is the target
X = df.iloc[:, :-1]
y = df.iloc[:, -1]

# 3. Check for missing values
print("\nMissing values in dataset:", df.isnull().sum().sum())

# 4. EDA
plt.figure(figsize=(6,4))
sns.countplot(x=y)
plt.title("Class Distribution (0 = Ham, 1 = Spam)")
plt.show()

# Basic histogram of a few features
X.iloc[:, :5].hist(bins=30, figsize=(12, 6))
plt.suptitle("Distribution of first few features")
plt.tight_layout()
plt.show()

# 5. Normalize features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# 6. Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, stratify=y, random_state=42)

# 7. Train Gaussian Naive Bayes
model = GaussianNB()
model.fit(X_train, y_train)

# 8. Evaluate on Test Set
y_pred = model.predict(X_test)

from sklearn.metrics import classification_report, confusion_matrix, accuracy_score


report = classification_report(y_test, y_pred, output_dict=True)

print("\n Classification Metrics (for class 1 - SPAM):")
print(f"Precision : {report['1']['precision']:.2f}")
print(f"Recall    : {report['1']['recall']:.2f}")
print(f"F1-score  : {report['1']['f1-score']:.2f}")
print(f"Accuracy  : {accuracy_score(y_test, y_pred):.2f}")

# Print confusion matrix
print("\n Confusion Matrix:")
cm = confusion_matrix(y_test, y_pred)
print(cm)

import seaborn as sns
```

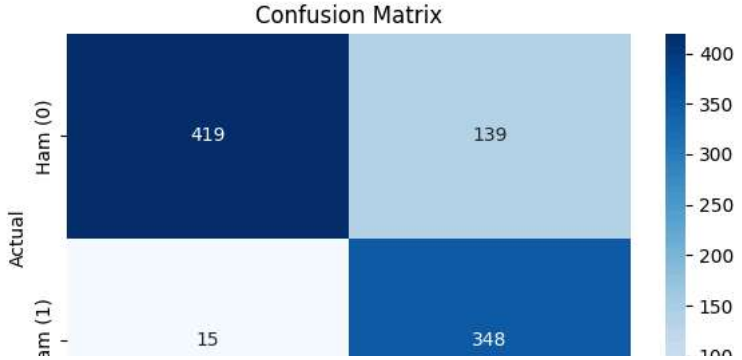
```
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix

# Generate confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Define class labels
labels = ['Ham (0)', 'Spam (1)']

# Plot using seaborn heatmap
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=labels, yticklabels=labels)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.tight_layout()
plt.show()


# ROC Curve
y_proba = model.predict_proba(X_test)[:, 1]
fpr, tpr, _ = roc_curve(y_test, y_proba)
roc_auc = auc(fpr, tpr)

plt.figure(figsize=(6,4))
plt.plot(fpr, tpr, label=f"ROC Curve (AUC = {roc_auc:.2f})")
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
plt.legend()
plt.show()

# 9. K-Fold Cross-Validation (K=5)
kf = KFold(n_splits=5, shuffle=True, random_state=42)
cv_accuracy = cross_val_score(model, X_scaled, y, cv=kf, scoring='accuracy')
cv_precision = cross_val_score(model, X_scaled, y, cv=kf, scoring='precision')
cv_recall = cross_val_score(model, X_scaled, y, cv=kf, scoring='recall')
cv_f1 = cross_val_score(model, X_scaled, y, cv=kf, scoring='f1')

print("\n--- 5-Fold Cross Validation Results ---")
print(f"Average Accuracy : {cv_accuracy.mean():.4f}")
print(f"Average Precision: {cv_precision.mean():.4f}")
print(f"Average Recall   : {cv_recall.mean():.4f}")
print(f"Average F1-Score : {cv_f1.mean():.4f}")
```

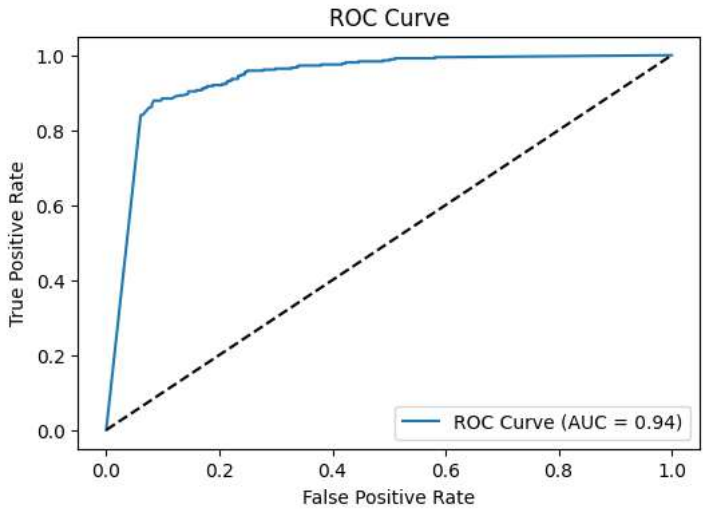
```
Gaussian (naive bayes)
Columns in dataset:
 ['word_freq_make', 'word_freq_address', 'word_freq_all', 'word_freq_3d', 'word_freq_our', 'word_freq_over', 'word_freq_remove', 'word_f
Dataset shape: (4601, 58)

Missing values in dataset: 0
```






```
 Classification Metrics (for class 1 - SPAM):
Precision : 0.71
Recall    : 0.96
F1-score  : 0.82
Accuracy  : 0.83

 Confusion Matrix:
[[419 139]
 [ 15 348]]
```



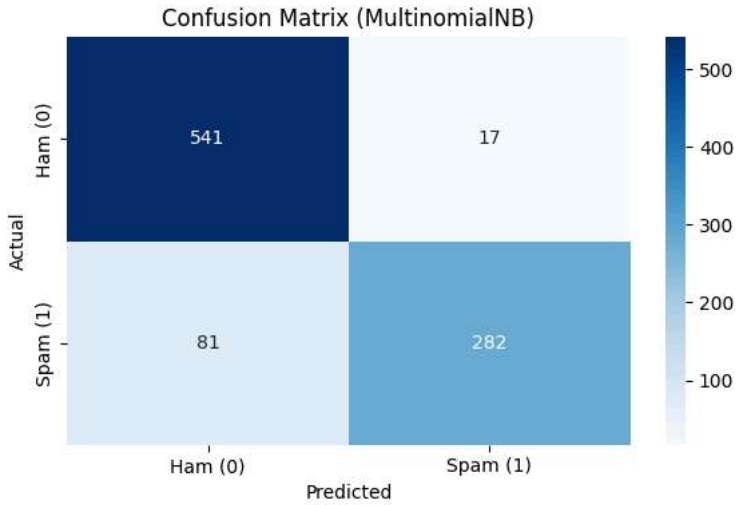


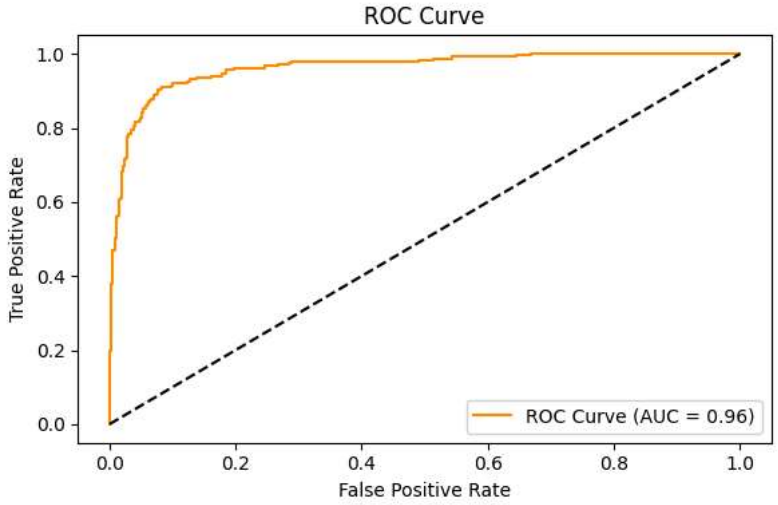

```
--- 5-Fold Cross Validation Results ---
Average Accuracy : 0.8153
Average Precision: 0.6933
Average Recall   : 0.9557
Average F1-Score : 0.8031
```

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, cross_val_score, KFold
from sklearn.naive_bayes import MultinomialNB
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, classification_report, confusion_matrix, roc_curve, auc
import matplotlib.pyplot as plt
import seaborn as sns

# 1. Load dataset
df = pd.read_csv('spambase_csv.csv')
print("Multinominal (naive Bayes)")
# 2. Split features and target
X = df.iloc[:, :-1]
y = df.iloc[:, -1]

# 3. Normalize features to [0, 1] range (MultinomialNB requires non-negative)
scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)

# 4. Train-test split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, stratify=y, random_state=42)

# 5. Train Multinomial Naive Bayes
model = MultinomialNB()
model.fit(X_train, y_train)

# 6. Predict
y_pred = model.predict(X_test)

# 7. Evaluation
report = classification_report(y_test, y_pred, output_dict=True)
print("\n Classification Metrics (for class 1 - SPAM):")
print(f"Precision : {report['1']['precision']:.2f}")
print(f"Recall    : {report['1']['recall']:.2f}")
print(f"F1-score  : {report['1']['f1-score']:.2f}")
print(f"Accuracy  : {accuracy_score(y_test, y_pred):.2f}")
```

```
# 8. Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
print("\nConfusion Matrix:")
print(cm)

# Plot Confusion Matrix
labels = ['Ham (0)', 'Spam (1)']
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=labels, yticklabels=labels)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix (MultinomialNB)')
plt.tight_layout()
plt.show()

# 9. ROC Curve
y_proba = model.predict_proba(X_test)[:, 1]
fpr, tpr, _ = roc_curve(y_test, y_proba)
roc_auc = auc(fpr, tpr)

plt.figure(figsize=(6, 4))
plt.plot(fpr, tpr, label=f"ROC Curve (AUC = {roc_auc:.2f})", color='darkorange')
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
plt.legend()
plt.tight_layout()
plt.show()

# 10. 5-Fold Cross-Validation
kf = KFold(n_splits=5, shuffle=True, random_state=42)
cv_accuracy = cross_val_score(model, X_scaled, y, cv=kf, scoring='accuracy')
cv_precision = cross_val_score(model, X_scaled, y, cv=kf, scoring='precision')
cv_recall = cross_val_score(model, X_scaled, y, cv=kf, scoring='recall')
cv_f1 = cross_val_score(model, X_scaled, y, cv=kf, scoring='f1')

print("\n--- 5-Fold Cross Validation Results ---")
print(f"Average Accuracy : {cv_accuracy.mean():.4f}")
print(f"Average Precision: {cv_precision.mean():.4f}")
print(f"Average Recall   : {cv_recall.mean():.4f}")
print(f"Average F1-Score : {cv_f1.mean():.4f}")
```

```
Multinominal (naive Bayes)

 Classification Metrics (for class 1 - SPAM):
Precision : 0.94
Recall    : 0.78
F1-score  : 0.85
Accuracy  : 0.89

Confusion Matrix:
[[541  17]
 [ 81 282]]
```



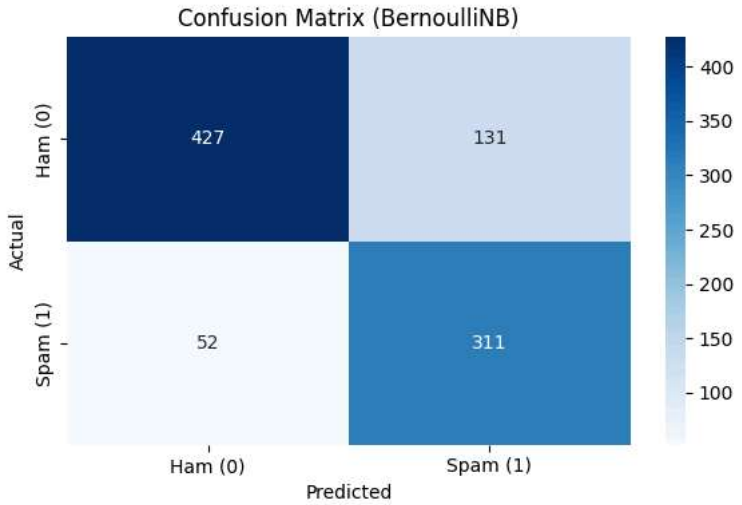


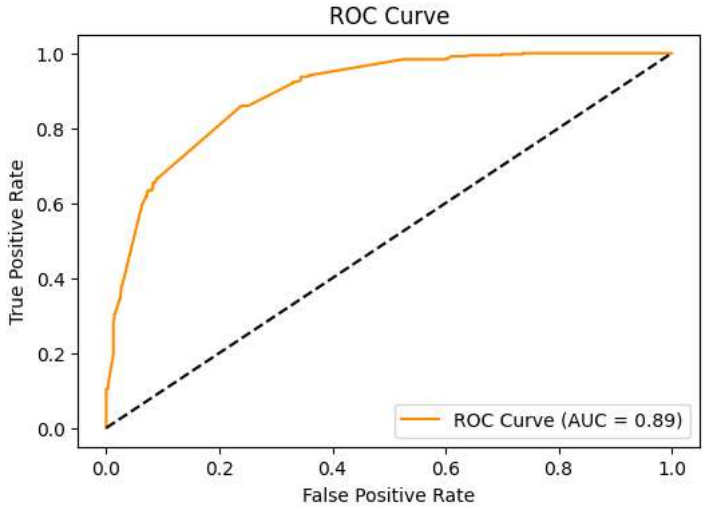

```
--- 5-Fold Cross Validation Results ---
Average Accuracy : 0.8863
Average Precision: 0.9364
Average Recall   : 0.7639
Average F1-Score : 0.8412
```

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import BernoulliNB
from sklearn.preprocessing import Binarizer
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix, classification_report
import seaborn as sns
import matplotlib.pyplot as plt

# 1. Load dataset
df = pd.read_csv('spambase_csv.csv')

# 2. Separate features and target
X = df.iloc[:, :-1]
y = df.iloc[:, -1]
print("Bernaulli (Naive Bayes)")

binarizer = Binarizer(threshold=X.median().mean())
```

```
X_bin = binarizer.fit_transform(X)

# 4. Train-test split
X_train, X_test, y_train, y_test = train_test_split(X_bin, y, test_size=0.2, stratify=y, random_state=42)

# 5. Train BernoulliNB
model = BernoulliNB()
model.fit(X_train, y_train)

# 6. Predict
y_pred = model.predict(X_test)

# 7. Evaluation
report = classification_report(y_test, y_pred, output_dict=True)
print("\n Classification Metrics (for class 1 - SPAM):")
print(f"Precision : {report['1']['precision']:.2f}")
print(f"Recall    : {report['1']['recall']:.2f}")
print(f"F1-score  : {report['1']['f1-score']:.2f}")
print(f"Accuracy  : {accuracy_score(y_test, y_pred):.2f}")

# 8. Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
print("\nConfusion Matrix:")
print(cm)

# Plot Confusion Matrix
labels = ['Ham (0)', 'Spam (1)']
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=labels, yticklabels=labels)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix (BernoulliNB)')
plt.tight_layout()
plt.show()


# 10. ROC Curve
y_proba = model.predict_proba(X_test)[:, 1]
fpr, tpr, _ = roc_curve(y_test, y_proba)
roc_auc = auc(fpr, tpr)

plt.figure(figsize=(6,4))
plt.plot(fpr, tpr, label=f"ROC Curve (AUC = {roc_auc:.2f})", color='darkorange')
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
plt.legend()
plt.show()

# 11. 5-Fold Cross-Validation
kf = KFold(n_splits=5, shuffle=True, random_state=42)
cv_accuracy = cross_val_score(model, X_bin, y, cv=kf, scoring='accuracy')
cv_precision = cross_val_score(model, X_bin, y, cv=kf, scoring='precision')
cv_recall = cross_val_score(model, X_bin, y, cv=kf, scoring='recall')
cv_f1 = cross_val_score(model, X_bin, y, cv=kf, scoring='f1')

print("\n--- 5-Fold Cross Validation Results ---")
print(f"Average Accuracy : {cv_accuracy.mean():.4f}")
print(f"Average Precision: {cv_precision.mean():.4f}")
print(f"Average Recall   : {cv_recall.mean():.4f}")
print(f"Average F1-Score : {cv_f1.mean():.4f}")
```

```
Bernaulli (Naive Bayes)

 Classification Metrics (for class 1 - SPAM):
Precision : 0.70
Recall    : 0.86
F1-score  : 0.77
Accuracy  : 0.80

Confusion Matrix:
[[427 131]
 [ 52 311]]
```



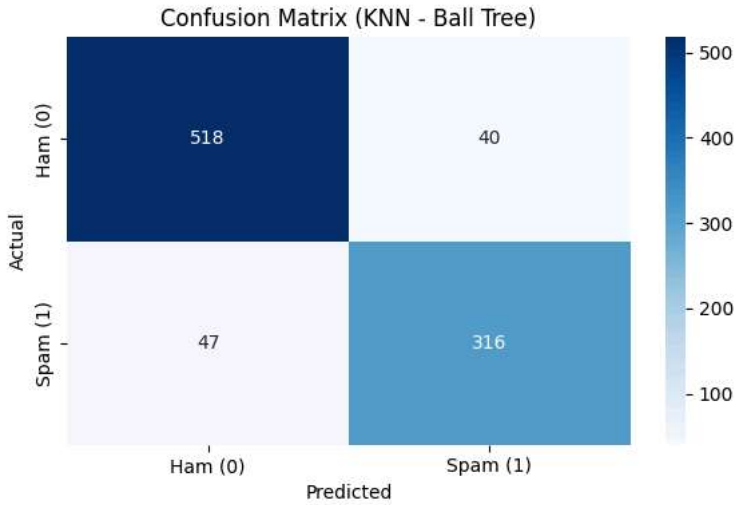


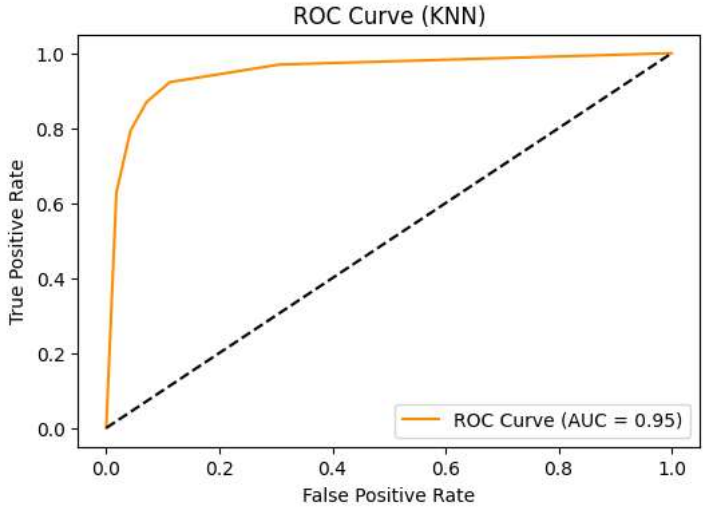

```
--- 5-Fold Cross Validation Results ---
Average Accuracy : 0.8040
Average Precision: 0.7028
Average Recall   : 0.8708
Average F1-Score : 0.7776
```

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, KFold, cross_val_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import (
    accuracy_score, precision_score, recall_score, f1_score,
    confusion_matrix, classification_report, roc_curve, auc
)
import seaborn as sns
import matplotlib.pyplot as plt

# 1. Load dataset
df = pd.read_csv('spambase_csv.csv')
print("Ball tree (KNN)")
# 2. Separate features and target
X = df.iloc[:, :-1]
y = df.iloc[:, -1]
```

```
# 3. Feature Scaling (important for KNN)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# 4. Train-test split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, stratify=y, random_state=42)

# 5. Train KNN with Ball Tree
model = KNeighborsClassifier(n_neighbors=5, algorithm='ball_tree')
model.fit(X_train, y_train)

# 6. Predict
y_pred = model.predict(X_test)

# 7. Evaluation
report = classification_report(y_test, y_pred, output_dict=True)
print("\nClassification Metrics (for class 1 - SPAM):")
print(f"Precision : {report['1']['precision']:.2f}")
print(f"Recall    : {report['1']['recall']:.2f}")
print(f"F1-score  : {report['1']['f1-score']:.2f}")
print(f"Accuracy  : {accuracy_score(y_test, y_pred):.2f}")

# 8. Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
print("\nConfusion Matrix:")
print(cm)

# Plot Confusion Matrix
labels = ['Ham (0)', 'Spam (1)']
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=labels, yticklabels=labels)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix (KNN - Ball Tree)')
plt.tight_layout()
plt.show()

# 9. ROC Curve
y_proba = model.predict_proba(X_test)[:, 1]
fpr, tpr, _ = roc_curve(y_test, y_proba)
roc_auc = auc(fpr, tpr)

plt.figure(figsize=(6,4))
plt.plot(fpr, tpr, label=f"ROC Curve (AUC = {roc_auc:.2f})", color='darkorange')
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve (KNN)")
plt.legend()
plt.show()

# 10. 5-Fold Cross Validation
kf = KFold(n_splits=5, shuffle=True, random_state=42)
cv_accuracy = cross_val_score(model, X_scaled, y, cv=kf, scoring='accuracy')
cv_precision = cross_val_score(model, X_scaled, y, cv=kf, scoring='precision')
cv_recall = cross_val_score(model, X_scaled, y, cv=kf, scoring='recall')
cv_f1 = cross_val_score(model, X_scaled, y, cv=kf, scoring='f1')

print("\n--- 5-Fold Cross Validation Results ---")
print(f"Average Accuracy : {cv_accuracy.mean():.4f}")
print(f"Average Precision: {cv_precision.mean():.4f}")
print(f"Average Recall   : {cv_recall.mean():.4f}")
print(f"Average F1-Score : {cv_f1.mean():.4f}")
```

```
Ball tree (KNN)

Classification Metrics (for class 1 - SPAM):
Precision : 0.89
Recall    : 0.87
F1-score  : 0.88
Accuracy  : 0.91

Confusion Matrix:
[[518  40]
 [ 47 316]]
```





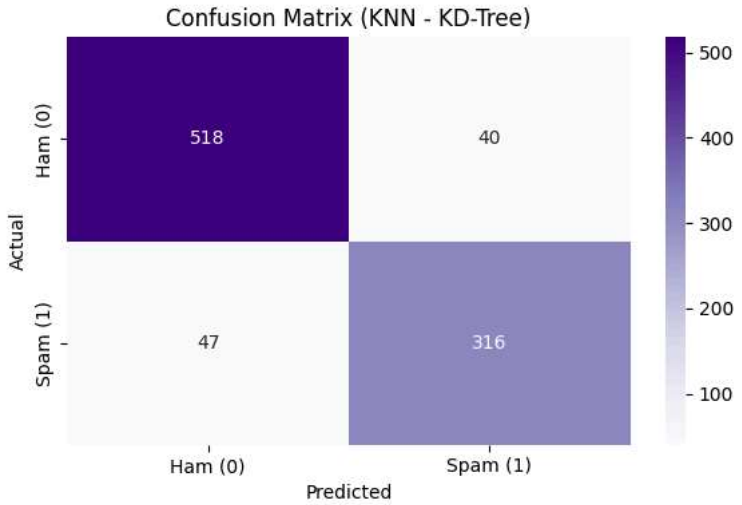

```
--- 5-Fold Cross Validation Results ---
Average Accuracy : 0.9085
Average Precision: 0.8983
Average Recall   : 0.8663
Average F1-Score : 0.8820
```

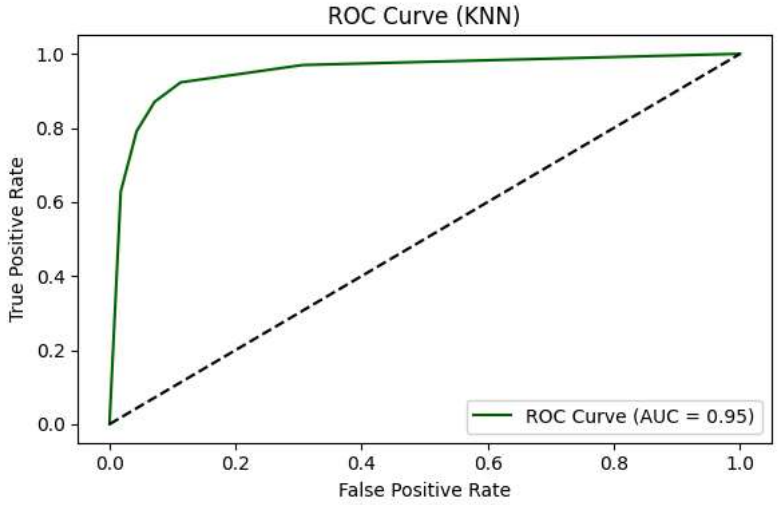
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, cross_val_score, KFold
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, classification_report, confusion_matrix, roc_curve, auc
import matplotlib.pyplot as plt
import seaborn as sns

# 1. Load dataset
df = pd.read_csv('spambase_csv.csv')

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

# 3. Standardize features (important for KNN)
scaler = StandardScaler()
```

```
X_scaled = scaler.fit_transform(X)
print("KD tree(KNN)")
# 4. Train-test split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, stratify=y, random_state=42)

# 5. KNN Classifier using KD-Tree
model = KNeighborsClassifier(n_neighbors=5, algorithm='kd_tree')  # You can tune k
model.fit(X_train, y_train)

# 6. Predict
y_pred = model.predict(X_test)

# 7. Evaluation
report = classification_report(y_test, y_pred, output_dict=True)
print("\n Classification Metrics (for class 1 - SPAM):")
print(f"Precision : {report['1']['precision']:.2f}")
print(f"Recall    : {report['1']['recall']:.2f}")
print(f"F1-score  : {report['1']['f1-score']:.2f}")
print(f"Accuracy  : {accuracy_score(y_test, y_pred):.2f}")

# 8. Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
print("\nConfusion Matrix:")
print(cm)

# Plot Confusion Matrix
labels = ['Ham (0)', 'Spam (1)']
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Purples', xticklabels=labels, yticklabels=labels)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix (KNN - KD-Tree)')
plt.tight_layout()
plt.show()

# 9. ROC Curve
if hasattr(model, "predict_proba"):
    y_proba = model.predict_proba(X_test)[:, 1]
    fpr, tpr, _ = roc_curve(y_test, y_proba)
    roc_auc = auc(fpr, tpr)

    plt.figure(figsize=(6, 4))
    plt.plot(fpr, tpr, label=f"ROC Curve (AUC = {roc_auc:.2f})", color='darkgreen')
    plt.plot([0, 1], [0, 1], 'k--')
    plt.xlabel("False Positive Rate")
    plt.ylabel("True Positive Rate")
    plt.title("ROC Curve (KNN)")
    plt.legend()
    plt.tight_layout()
    plt.show()

# 10. 5-Fold Cross-Validation
kf = KFold(n_splits=5, shuffle=True, random_state=42)
cv_accuracy = cross_val_score(model, X_scaled, y, cv=kf, scoring='accuracy')
cv_precision = cross_val_score(model, X_scaled, y, cv=kf, scoring='precision')
cv_recall = cross_val_score(model, X_scaled, y, cv=kf, scoring='recall')
cv_f1 = cross_val_score(model, X_scaled, y, cv=kf, scoring='f1')

print("\n--- 5-Fold Cross Validation Results ---")
print(f"Average Accuracy : {cv_accuracy.mean():.4f}")
print(f"Average Precision: {cv_precision.mean():.4f}")
print(f"Average Recall   : {cv_recall.mean():.4f}")
print(f"Average F1-Score : {cv_f1.mean():.4f}")
```

KD tree(KNN)

```
 Classification Metrics (for class 1 - SPAM):
Precision : 0.89
Recall    : 0.87
F1-score  : 0.88
Accuracy  : 0.91

Confusion Matrix:
[[518  40]
 [ 47 316]]
```






```
--- 5-Fold Cross Validation Results ---
Average Accuracy : 0.9085
Average Precision: 0.8983
Average Recall   : 0.8663
Average F1-Score : 0.8820
```

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, cross_val_score, KFold
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, classification_report, confusion_matrix, roc_curve, au
import matplotlib.pyplot as plt
import seaborn as sns

# 1. Load dataset
df = pd.read_csv('spambase_csv.csv')

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

# 3. Standardize features
scaler = StandardScaler()
```