naive

August 5, 2025

0.0.1 Importing Required Libraries

[19]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler  # or MinMaxScaler, if used
from sklearn.preprocessing import LabelEncoder  # if categorical labels
from sklearn.naive_bayes import BernoulliNB, MultinomialNB, GaussianNB
from sklearn.metrics import accuracy_score, confusion_matrix,␣
↪classification_report
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.model_selection import cross_val_score, KFold
```

0.0.2 Loading dataset

[2]:
```
df = pd.read_csv('spambase.csv')
```

0.0.3 Basic info

[3]:
```
print("Dataset Info:\n", df.info())
print("\nFirst 5 rows:\n", df.head())
print("\nMissing Values:\n", df.isnull().sum())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4601 entries, 0 to 4600
Data columns (total 58 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   word_freq_make          4601 non-null   float64
 1   word_freq_address       4601 non-null   float64
 2   word_freq_all           4601 non-null   float64
 3   word_freq_3d            4601 non-null   float64
 4   word_freq_our           4601 non-null   float64
 5   word_freq_over          4601 non-null   float64
 6   word_freq_remove        4601 non-null   float64
```

```
7   word_freq_internet          4601 non-null   float64
8   word_freq_order             4601 non-null   float64
9   word_freq_mail              4601 non-null   float64
10  word_freq_receive           4601 non-null   float64
11  word_freq_will              4601 non-null   float64
12  word_freq_people            4601 non-null   float64
13  word_freq_report            4601 non-null   float64
14  word_freq_addresses         4601 non-null   float64
15  word_freq_free              4601 non-null   float64
16  word_freq_business          4601 non-null   float64
17  word_freq_email             4601 non-null   float64
18  word_freq_you               4601 non-null   float64
19  word_freq_credit            4601 non-null   float64
20  word_freq_your              4601 non-null   float64
21  word_freq_font              4601 non-null   float64
22  word_freq_000               4601 non-null   float64
23  word_freq_money             4601 non-null   float64
24  word_freq_hp                4601 non-null   float64
25  word_freq_hpl               4601 non-null   float64
26  word_freq_george            4601 non-null   float64
27  word_freq_650               4601 non-null   float64
28  word_freq_lab               4601 non-null   float64
29  word_freq_labs              4601 non-null   float64
30  word_freq_telnet            4601 non-null   float64
31  word_freq_857               4601 non-null   float64
32  word_freq_data              4601 non-null   float64
33  word_freq_415               4601 non-null   float64
34  word_freq_85                4601 non-null   float64
35  word_freq_technology        4601 non-null   float64
36  word_freq_1999              4601 non-null   float64
37  word_freq_parts             4601 non-null   float64
38  word_freq_pm                4601 non-null   float64
39  word_freq_direct            4601 non-null   float64
40  word_freq_cs                4601 non-null   float64
41  word_freq_meeting           4601 non-null   float64
42  word_freq_original          4601 non-null   float64
43  word_freq_project           4601 non-null   float64
44  word_freq_re                4601 non-null   float64
45  word_freq_edu               4601 non-null   float64
46  word_freq_table             4601 non-null   float64
47  word_freq_conference        4601 non-null   float64
48  char_freq_%3B               4601 non-null   float64
49  char_freq_%28               4601 non-null   float64
50  char_freq_%5B               4601 non-null   float64
51  char_freq_%21               4601 non-null   float64
52  char_freq_%24               4601 non-null   float64
53  char_freq_%23               4601 non-null   float64
54  capital_run_length_average  4601 non-null   float64
```

```
 55  capital_run_length_longest  4601 non-null   int64
 56  capital_run_length_total    4601 non-null   int64
 57  class                       4601 non-null   int64
dtypes: float64(55), int64(3)
memory usage: 2.0 MB
Dataset Info:
 None

First 5 rows:
    word_freq_make  word_freq_address  word_freq_all  word_freq_3d  \
0            0.00               0.64           0.64           0.0
1            0.21               0.28           0.50           0.0
2            0.06               0.00           0.71           0.0
3            0.00               0.00           0.00           0.0
4            0.00               0.00           0.00           0.0

   word_freq_our  word_freq_over  word_freq_remove  word_freq_internet  \
0           0.32            0.00              0.00                0.00
1           0.14            0.28              0.21                0.07
2           1.23            0.19              0.19                0.12
3           0.63            0.00              0.31                0.63
4           0.63            0.00              0.31                0.63

   word_freq_order  word_freq_mail  …  char_freq_%3B  char_freq_%28  \
0             0.00            0.00  …           0.00          0.000
1             0.00            0.94  …           0.00          0.132
2             0.64            0.25  …           0.01          0.143
3             0.31            0.63  …           0.00          0.137
4             0.31            0.63  …           0.00          0.135

   char_freq_%5B  char_freq_%21  char_freq_%24  char_freq_%23  \
0            0.0          0.778          0.000          0.000
1            0.0          0.372          0.180          0.048
2            0.0          0.276          0.184          0.010
3            0.0          0.137          0.000          0.000
4            0.0          0.135          0.000          0.000

   capital_run_length_average  capital_run_length_longest  \
0                       3.756                          61
1                       5.114                         101
2                       9.821                         485
3                       3.537                          40
4                       3.537                          40

   capital_run_length_total  class
0                       278      1
1                      1028      1
2                      2259      1
```

```
3                       191      1
4                       191      1

[5 rows x 58 columns]

Missing Values:
 word_freq_make               0
word_freq_address             0
word_freq_all                 0
word_freq_3d                  0
word_freq_our                 0
word_freq_over                0
word_freq_remove              0
word_freq_internet            0
word_freq_order               0
word_freq_mail                0
word_freq_receive             0
word_freq_will                0
word_freq_people              0
word_freq_report              0
word_freq_addresses           0
word_freq_free                0
word_freq_business            0
word_freq_email               0
word_freq_you                 0
word_freq_credit              0
word_freq_your                0
word_freq_font                0
word_freq_000                 0
word_freq_money               0
word_freq_hp                  0
word_freq_hpl                 0
word_freq_george              0
word_freq_650                 0
word_freq_lab                 0
word_freq_labs                0
word_freq_telnet              0
word_freq_857                 0
word_freq_data                0
word_freq_415                 0
word_freq_85                  0
word_freq_technology          0
word_freq_1999                0
word_freq_parts               0
word_freq_pm                  0
word_freq_direct              0
word_freq_cs                  0
word_freq_meeting             0
```

```
word_freq_original            0
word_freq_project             0
word_freq_re                  0
word_freq_edu                 0
word_freq_table               0
word_freq_conference          0
char_freq_%3B                 0
char_freq_%28                 0
char_freq_%5B                 0
char_freq_%21                 0
char_freq_%24                 0
char_freq_%23                 0
capital_run_length_average    0
capital_run_length_longest    0
capital_run_length_total      0
class                         0
dtype: int64
```

0.0.4 Handling missing values

[6]:
```
imputer = SimpleImputer(strategy='mean')
df_imputed = pd.DataFrame(imputer.fit_transform(df), columns=df.columns)
```

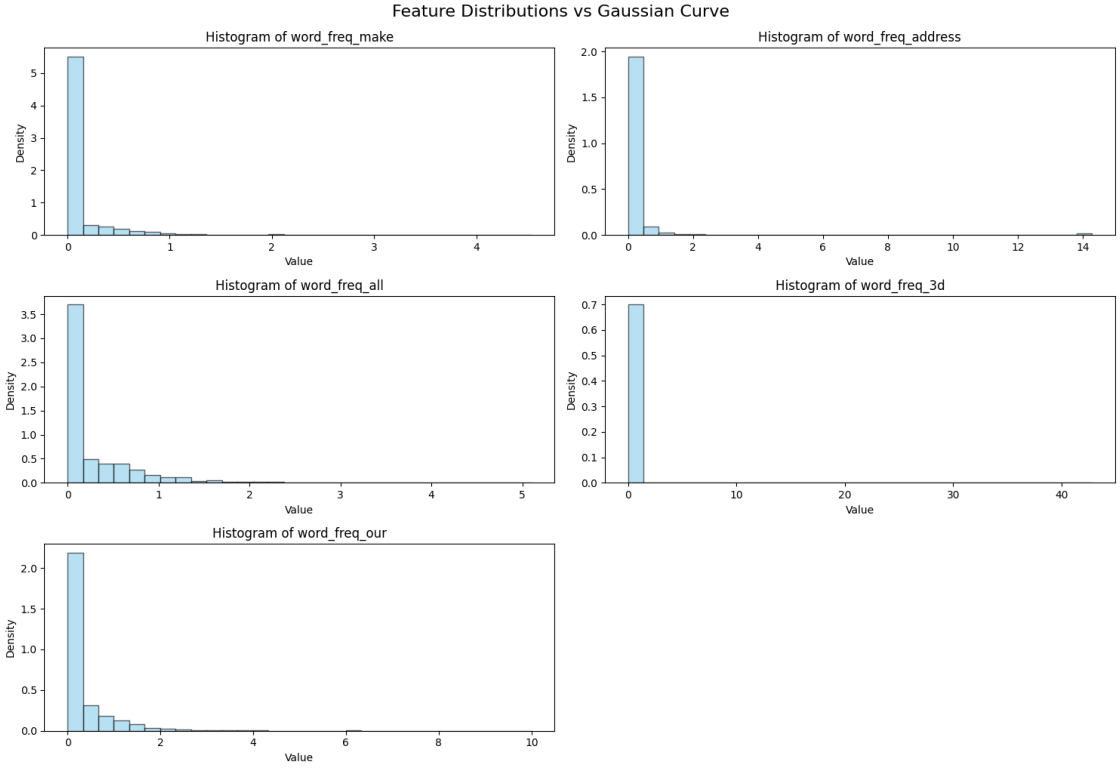
0.0.5 Splitting of feature and target

[7]:
```
X = df_imputed.drop('class', axis=1)
y = df_imputed['class']
```

0.0.6 Checking Distribution

[8]:
```
X = df.drop('class', axis=1)  # Assuming 'class' is the target

# Choose a few features to visualize
sample_features = X.columns[:5]  # First 5 features

# Plot histograms with Gaussian curve overlay
plt.figure(figsize=(15, 10))
for i, feature in enumerate(sample_features):
    plt.subplot(3, 2, i + 1)
    data = X[feature]

    # Plot histogram
    count, bins, ignored = plt.hist(data, bins=30, density=True, alpha=0.6,␣
 ↪color='skyblue', edgecolor='black')

    # Plot normal distribution curve
    '''mu, std = data.mean(), data.std()
```

```
    xmin, xmax = plt.xlim()
    x = np.linspace(xmin, xmax, 100)
    p = 1 / (std * np.sqrt(2 * np.pi)) * np.exp(-(x - mu)**2 / (2 * std**2))
    plt.plot(x, p, 'r', linewidth=2)'''

    plt.title(f'Histogram of {feature}')
    plt.xlabel('Value')
    plt.ylabel('Density')

plt.tight_layout()
plt.suptitle('Feature Distributions vs Gaussian Curve', fontsize=16, y=1.02)
plt.show()
```



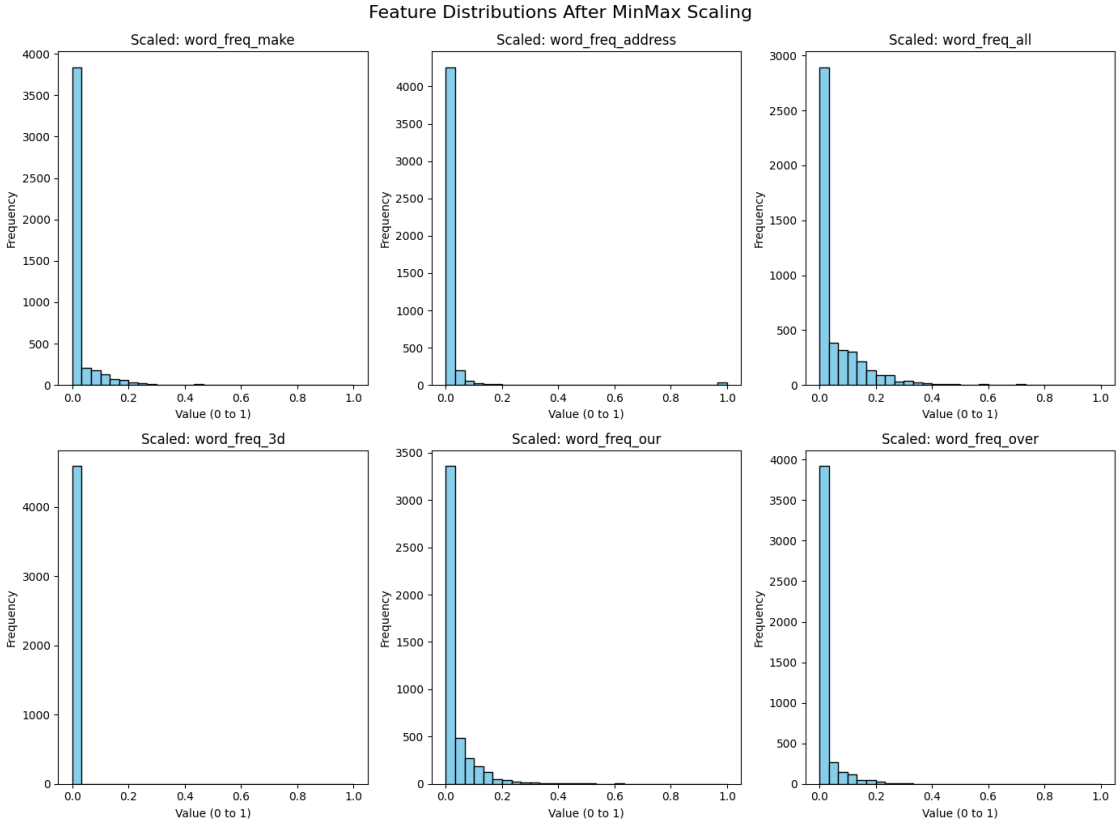
0.0.7 Applying min max scaling

[10]:
```
scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)
```

0.1 Plots

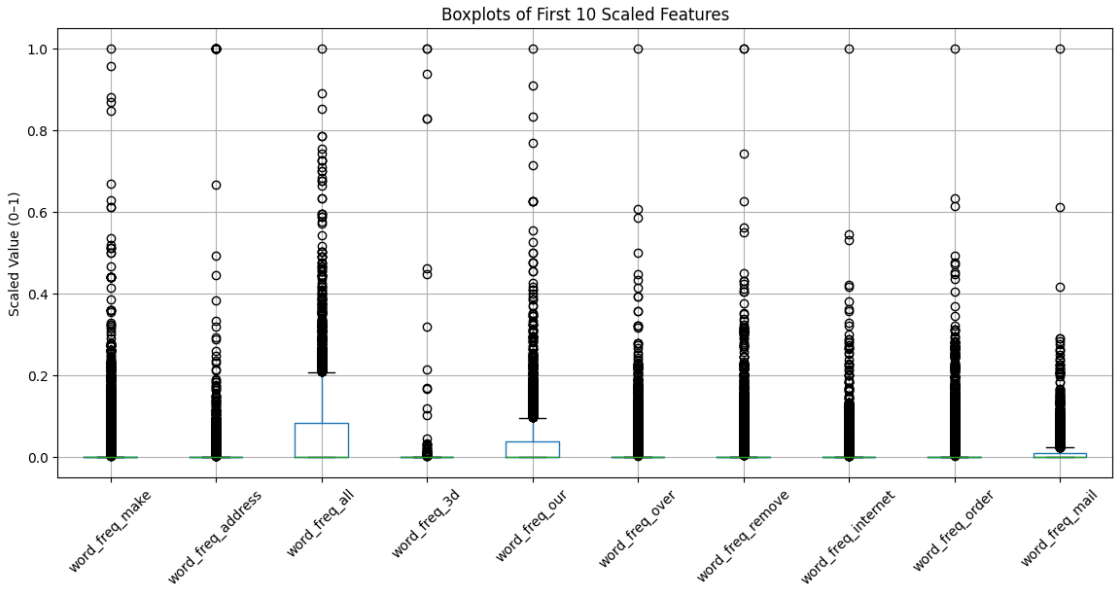
0.1.1 Histogram

[11]:
```
import matplotlib.pyplot as plt

plt.figure(figsize=(14, 10))
for i in range(6):  # first 6 features as an example
    plt.subplot(2, 3, i + 1)
    plt.hist(X_scaled[:, i], bins=30, color='skyblue', edgecolor='black')
    plt.title(f'Scaled: {X.columns[i]}')
    plt.xlabel('Value (0 to 1)')
    plt.ylabel('Frequency')
plt.tight_layout()
plt.suptitle('Feature Distributions After MinMax Scaling', fontsize=16, y=1.02)
plt.show()
```



0.1.2 Boxplot

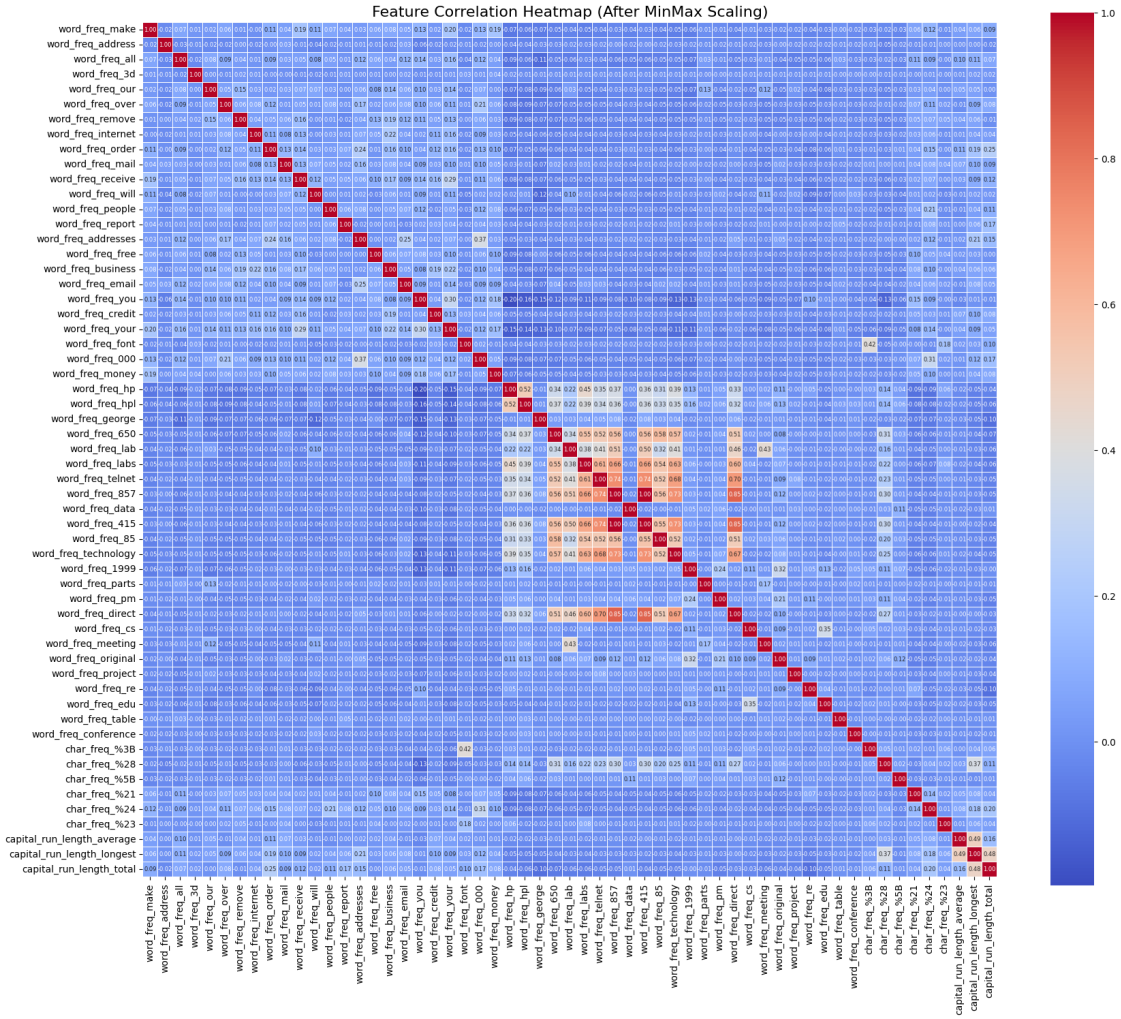
[12]:
```
import pandas as pd

# Convert scaled array back to DataFrame for easier plotting
X_scaled_df = pd.DataFrame(X_scaled, columns=X.columns)

plt.figure(figsize=(14, 6))
X_scaled_df.iloc[:, :10].boxplot(rot=45)
plt.title('Boxplots of First 10 Scaled Features')
plt.ylabel('Scaled Value (0-1)')
plt.grid(True)
plt.show()
```



0.1.3 Correlation HeatMap

[13]:
```
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

# Assuming X_scaled_df is your scaled DataFrame
corr_matrix = X_scaled_df.corr()

plt.figure(figsize=(18, 15))  # Bigger figure
sns.heatmap(
    corr_matrix,
    cmap='coolwarm',
```

```
    square=True,
    annot=True,                # Show values inside squares
    fmt=".2f",                 # Format to 2 decimal places
    linewidths=0.5,            # Thin grid lines
    annot_kws={"size": 6}      # Smaller font size
)

plt.xticks(rotation=90)
plt.yticks(rotation=0)
plt.title("Feature Correlation Heatmap (After MinMax Scaling)", fontsize=16)
plt.tight_layout()
plt.show()
```



0.1.4 Model Training

[14]:
```
from sklearn.metrics import (
    accuracy_score, precision_score, recall_score,
    f1_score, confusion_matrix, roc_curve, auc,
    classification_report
)

# Split the data again (or reuse your earlier split)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, test_size=0.3, stratify=y, random_state=42
)
```

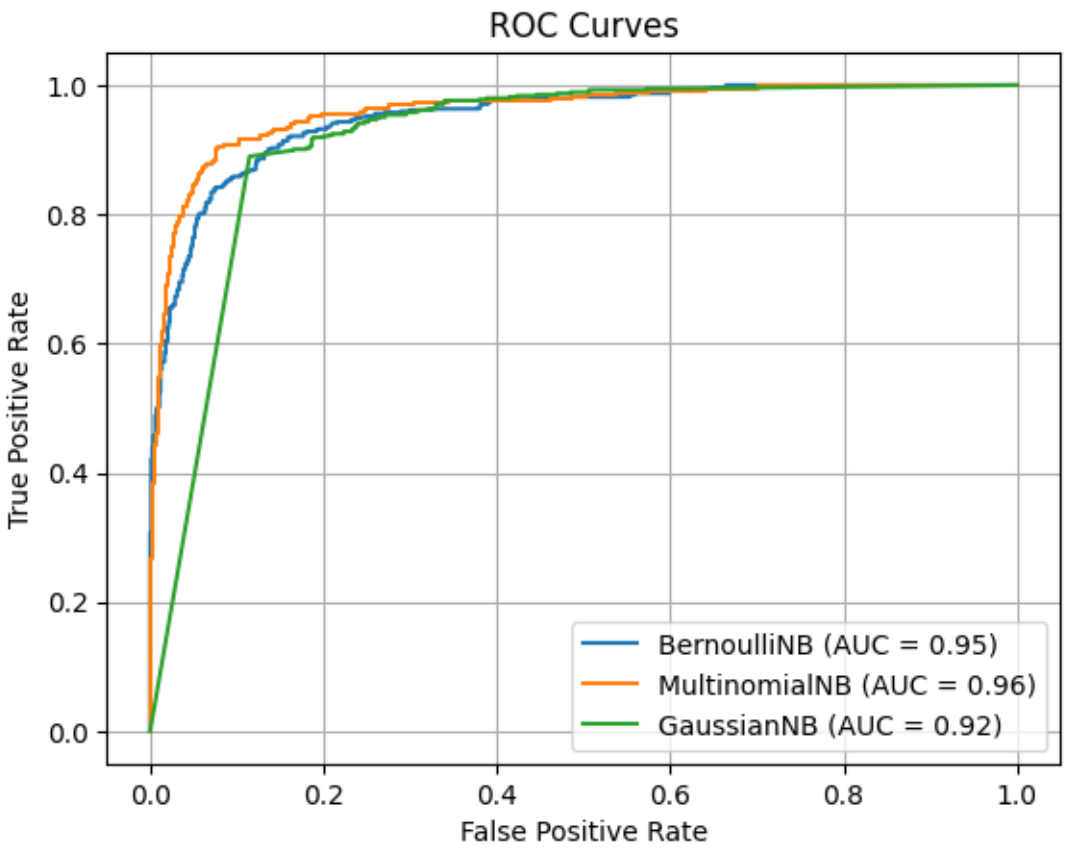
[16]:
```
def evaluate_model(name, model, X_test, y_test, results):
    y_pred = model.predict(X_test)

    # For ROC and AUC, use predict_proba
    try:
        y_proba = model.predict_proba(X_test)[:, 1]
    except:
        y_proba = y_pred  # fallback if proba not available

    acc = accuracy_score(y_test, y_pred)
    prec = precision_score(y_test, y_pred, zero_division=0)
    rec = recall_score(y_test, y_pred, zero_division=0)
    f1 = f1_score(y_test, y_pred, zero_division=0)

    # Print Confusion Matrix and Classification Report
    print(f"\n=== {name} ===")
    print("Confusion Matrix:")
    print(confusion_matrix(y_test, y_pred))
    print("\nClassification Report:")
    print(classification_report(y_test, y_pred, zero_division=0))

    # ROC and AUC
    fpr, tpr, _ = roc_curve(y_test, y_proba)
    roc_auc = auc(fpr, tpr)

    plt.plot(fpr, tpr, label=f'{name} (AUC = {roc_auc:.2f})')

    # Append results
    results.append({
        'Model': name,
        'Accuracy': acc,
        'Precision': prec,
        'Recall': rec,
```

```
        'F1 Score': f1,
        'AUC': roc_auc
    })
```

[17]:
```
# Store results
results = []

# --- BernoulliNB ---
bnb = BernoulliNB()
bnb.fit(X_train, y_train)
evaluate_model("BernoulliNB", bnb, X_test, y_test, results)

# --- MultinomialNB ---
mnb = MultinomialNB()
mnb.fit(X_train, y_train)
evaluate_model("MultinomialNB", mnb, X_test, y_test, results)

# --- GaussianNB ---
gnb = GaussianNB()
gnb.fit(X_train, y_train)
evaluate_model("GaussianNB", gnb, X_test, y_test, results)

# === ROC Curve Plot ===
plt.title('ROC Curves')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc='lower right')
plt.grid(True)
plt.show()

# === Final Comparison Table ===
results_df = pd.DataFrame(results)
print("\n=== Comparison Table ===")
print(results_df)
```

```
=== BernoulliNB ===
Confusion Matrix:
[[778  59]
 [ 93 451]]

Classification Report:
              precision    recall  f1-score   support

         0.0       0.89      0.93      0.91       837
         1.0       0.88      0.83      0.86       544

    accuracy                           0.89      1381
```

```
   macro avg       0.89      0.88      0.88      1381
weighted avg       0.89      0.89      0.89      1381


=== MultinomialNB ===
Confusion Matrix:
[[811  26]
 [119 425]]

Classification Report:
              precision    recall  f1-score   support

         0.0       0.87      0.97      0.92       837
         1.0       0.94      0.78      0.85       544

    accuracy                           0.90      1381
   macro avg       0.91      0.88      0.89      1381
weighted avg       0.90      0.90      0.89      1381


=== GaussianNB ===
Confusion Matrix:
[[616 221]
 [ 28 516]]

Classification Report:
              precision    recall  f1-score   support

         0.0       0.96      0.74      0.83       837
         1.0       0.70      0.95      0.81       544

    accuracy                           0.82      1381
   macro avg       0.83      0.84      0.82      1381
weighted avg       0.86      0.82      0.82      1381
```




```
=== Comparison Table ===
           Model  Accuracy  Precision    Recall  F1 Score       AUC
0    BernoulliNB  0.889935   0.884314  0.829044  0.855787  0.950023
1  MultinomialNB  0.895004   0.942350  0.781250  0.854271  0.960696
2     GaussianNB  0.819696   0.700136  0.948529  0.805621  0.915675
```

[20]:
```python
cv = KFold(n_splits=5, shuffle=True, random_state=42)

# Initialize Multinomial Naive Bayes
mnb = MultinomialNB()

# Perform cross-validation (accuracy as scoring)
scores = cross_val_score(mnb, X_scaled, y, cv=cv, scoring='accuracy')

# Print accuracy for each fold
print("  Multinomial Naive Bayes - 5-Fold Cross-Validation Results:")
for i, score in enumerate(scores, 1):
    print(f"Fold {i} Accuracy: {score:.4f}")
```

```python
# Print average and standard deviation
print(f"\nMean Accuracy        : {scores.mean():.4f}")
print(f"Standard Deviation   : {scores.std():.4f}")
```

```
 Multinomial Naive Bayes - 5-Fold Cross-Validation Results:
Fold 1 Accuracy: 0.8719
Fold 2 Accuracy: 0.8935
Fold 3 Accuracy: 0.8891
Fold 4 Accuracy: 0.8913
Fold 5 Accuracy: 0.8859

Mean Accuracy        : 0.8863
Standard Deviation   : 0.0077
```