

In [1]: `!pip install pandas numpy matplotlib scikit-learn`

```
Requirement already satisfied: pandas in c:\users\abhiram\appdata\local\programs\python\python313\lib\site-packages (2.3.3)
Requirement already satisfied: numpy in c:\users\abhiram\appdata\local\programs\python\python313\lib\site-packages (2.1.3)
Requirement already satisfied: matplotlib in c:\users\abhiram\appdata\local\programs\python\python313\lib\site-packages (3.10.0)
Requirement already satisfied: scikit-learn in c:\users\abhiram\appdata\local\programs\python\python313\lib\site-packages (1.7.2)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\abhiram\appdata\local\programs\python\python313\lib\site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\abhiram\appdata\local\programs\python\python313\lib\site-packages (from pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in c:\users\abhiram\appdata\local\programs\python\python313\lib\site-packages (from pandas) (2025.2)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\abhiram\appdata\local\programs\python\python313\lib\site-packages (from matplotlib) (1.3.3)
Requirement already satisfied: cycler>=0.10 in c:\users\abhiram\appdata\local\programs\python\python313\lib\site-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\abhiram\appdata\local\programs\python\python313\lib\site-packages (from matplotlib) (4.60.1)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\abhiram\appdata\local\programs\python\python313\lib\site-packages (from matplotlib) (1.4.9)
Requirement already satisfied: packaging>=20.0 in c:\users\abhiram\appdata\local\programs\python\python313\lib\site-packages (from matplotlib) (25.0)
Requirement already satisfied: pillow>=8 in c:\users\abhiram\appdata\local\programs\python\python313\lib\site-packages (from matplotlib) (11.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\abhiram\appdata\local\programs\python\python313\lib\site-packages (from matplotlib) (3.2.5)
Requirement already satisfied: scipy>=1.8.0 in c:\users\abhiram\appdata\local\programs\python\python313\lib\site-packages (from scikit-learn) (1.15.3)
Requirement already satisfied: joblib>=1.2.0 in c:\users\abhiram\appdata\local\programs\python\python313\lib\site-packages (from scikit-learn) (1.5.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in c:\users\abhiram\appdata\local\programs\python\python313\lib\site-packages (from scikit-learn) (3.6.0)
Requirement already satisfied: six>=1.5 in c:\users\abhiram\appdata\local\programs\python\python313\lib\site-packages (from python-dateutil>=2.8.2->pandas) (1.17.0)
```

[notice] A new release of pip is available: 25.2 -> 25.3

[notice] To update, run: python.exe -m pip install --upgrade pip

```
In [24]: import os
import re
import string
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB, GaussianNB
from sklearn.metrics import (
    accuracy_score, precision_score, recall_score, f1_score,
    confusion_matrix, ConfusionMatrixDisplay,
    roc_curve, auc, classification_report
)
```

```
In [26]: df = pd.read_csv(r"C:\Users\ABHIRAM\Downloads\Spam-Email-Detection-Project\datas

df = df[['v1', 'v2']]
df.columns = ['label', 'text']

df.head()
```

```
Out[26]:
```

	label	text
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

```
In [27]: # Convert label column to string safely
df["label"] = df["label"].astype(str)

# Normalize labels
df["label"] = df["label"].str.strip().str.lower()

# Keep only valid labels
df = df[df["label"].isin(["ham", "spam"])]

# Encode labels
df["label"] = df["label"].map({"ham": 0, "spam": 1})

print("Unique labels:", df["label"].unique())
print("Any NaN labels:", df["label"].isna().sum())
```

```
Unique labels: [0 1]
Any NaN labels: 0
```

```
In [28]: STOPWORDS = {
    "a", "an", "the", "and", "or", "but", "if", "while", "with", "to", "from", "of", "in", "c
    "for", "at", "by", "is", "are", "was", "were", "be", "been", "this", "that", "it", "as",
    "i", "you", "he", "she", "we", "they", "me", "my", "your", "our", "their", "so", "do", "d
}

def clean_text(text):
    text = str(text).lower()
    text = re.sub(r"http\S+|www\S+|https\S+", "", text)
    text = re.sub(r"\d+", "", text)
    text = text.translate(str.maketrans("", "", string.punctuation))
    text = re.sub(r"\s+", " ", text).strip()
    return " ".join([w for w in text.split() if w not in STOPWORDS])

df["clean_text"] = df["text"].apply(clean_text)

df.head()
```

Out[28]:

	label	text	clean_text
0	0	Go until jurong point, crazy.. Available only ...	go until jurong point crazy available only bug...
1	0	Ok lar... Joking wif u oni...	ok lar joking wif u oni
2	1	Free entry in 2 a wkly comp to win FA Cup fina...	free entry wkly comp win fa cup final tkts st ...
3	0	U dun say so early hor... U c already then say...	u dun say early hor u c already then say
4	0	Nah I don't think he goes to usf, he lives aro...	nah dont think goes usf lives around here though

```
In [29]: X = df["clean_text"].values
y = df["label"].values

X_train, X_test, y_train, y_test = train_test_split(
    X,
    y,
    test_size=0.2,
    random_state=42,
    stratify=y
)

print("Train size:", len(X_train))
print("Test size:", len(X_test))
```

Train size: 4457

Test size: 1115

```
In [30]: def evaluate_model(model, model_type, X_test_vec, y_test, title):
    if model_type == "gaussian":
        y_pred = model.predict(X_test_vec.toarray())
        y_prob = model.predict_proba(X_test_vec.toarray())[:, 1]
    else:
        y_pred = model.predict(X_test_vec)
        y_prob = model.predict_proba(X_test_vec)[:, 1]

    print("\n", title)
    print(classification_report(y_test, y_pred))

    # Confusion Matrix
    cm = confusion_matrix(y_test, y_pred)
    ConfusionMatrixDisplay(cm, display_labels=["Ham", "Spam"]).plot()
    plt.title(title + " - Confusion Matrix")
    plt.show()

    # ROC Curve
    fpr, tpr, _ = roc_curve(y_test, y_prob)
    roc_auc = auc(fpr, tpr)

    plt.plot(fpr, tpr, label=f"AUC = {roc_auc:.3f}")
    plt.plot([0,1], [0,1], "--")
    plt.xlabel("False Positive Rate")
    plt.ylabel("True Positive Rate")
    plt.title(title + " - ROC Curve")
    plt.legend()
```

```
plt.show()

return {
    "Model": title,
    "Accuracy": accuracy_score(y_test, y_pred),
    "Precision": precision_score(y_test, y_pred),
    "Recall": recall_score(y_test, y_pred),
    "F1": f1_score(y_test, y_pred)
}
```

```
In [31]: results = []

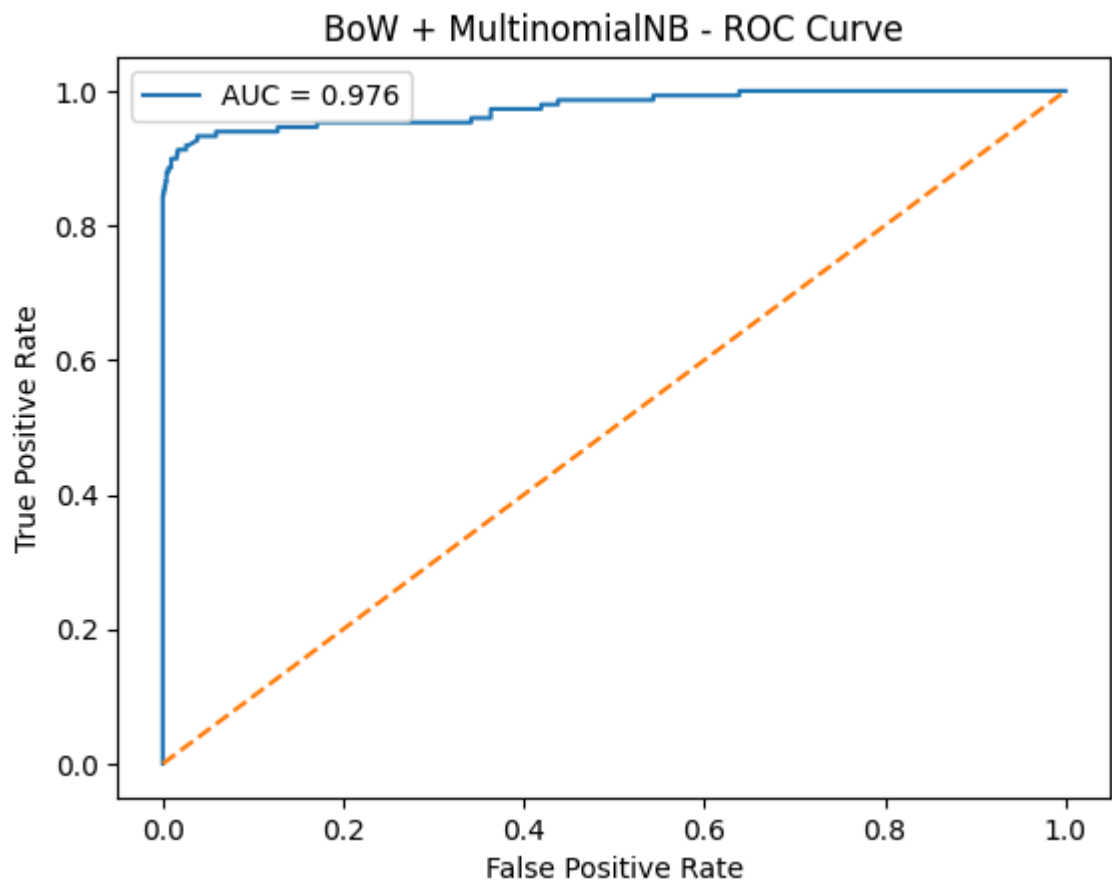
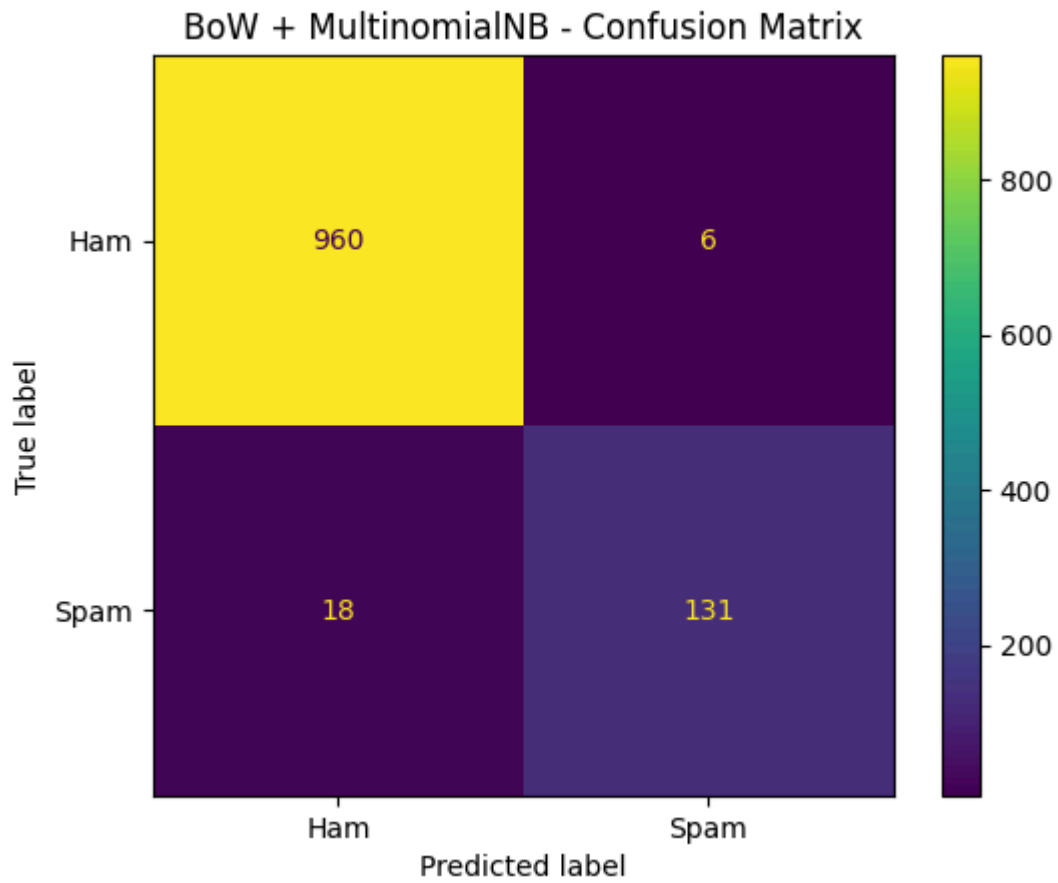
vectorizers = {
    "BoW": CountVectorizer(max_features=5000),
    "TF-IDF": TfidfVectorizer(max_features=5000)
}

for name, vectorizer in vectorizers.items():
    X_train_vec = vectorizer.fit_transform(X_train)
    X_test_vec = vectorizer.transform(X_test)

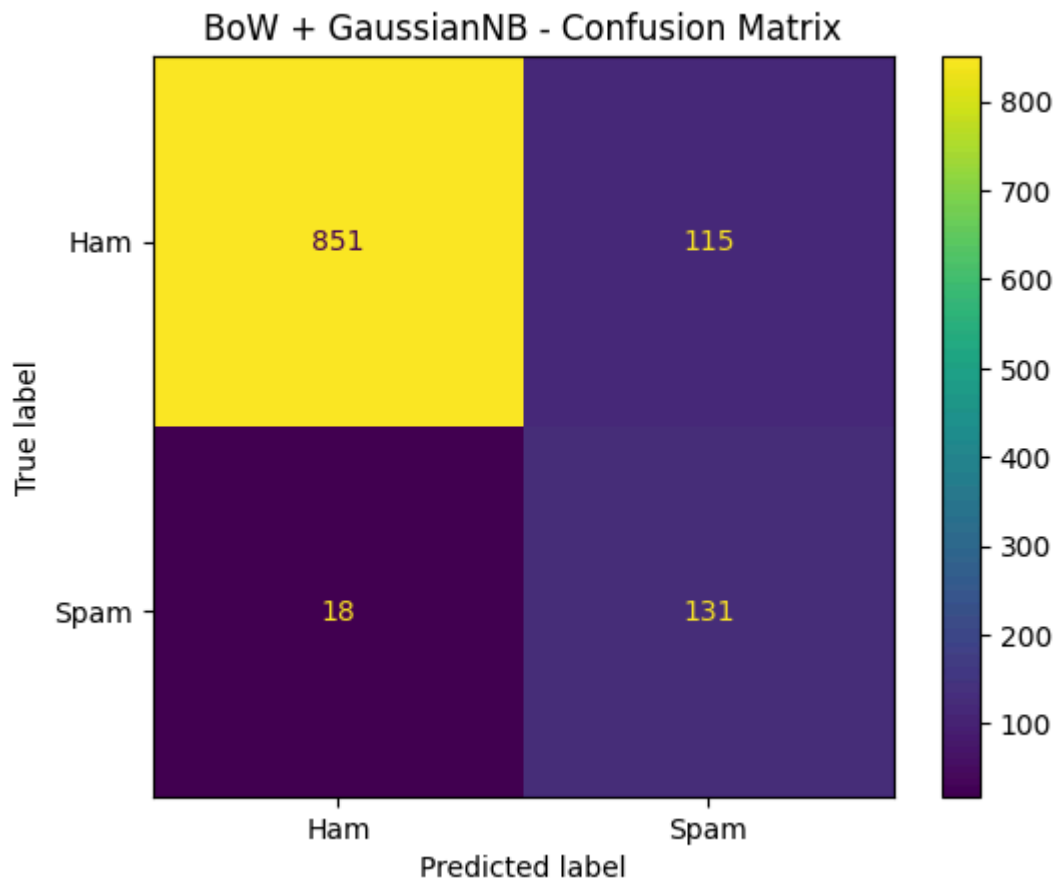
    # Multinomial NB
    mnb = MultinomialNB()
    mnb.fit(X_train_vec, y_train)
    results.append(
        evaluate_model(
            mnb,
            "multinomial",
            X_test_vec,
            y_test,
            f"{name} + MultinomialNB"
        )
    )

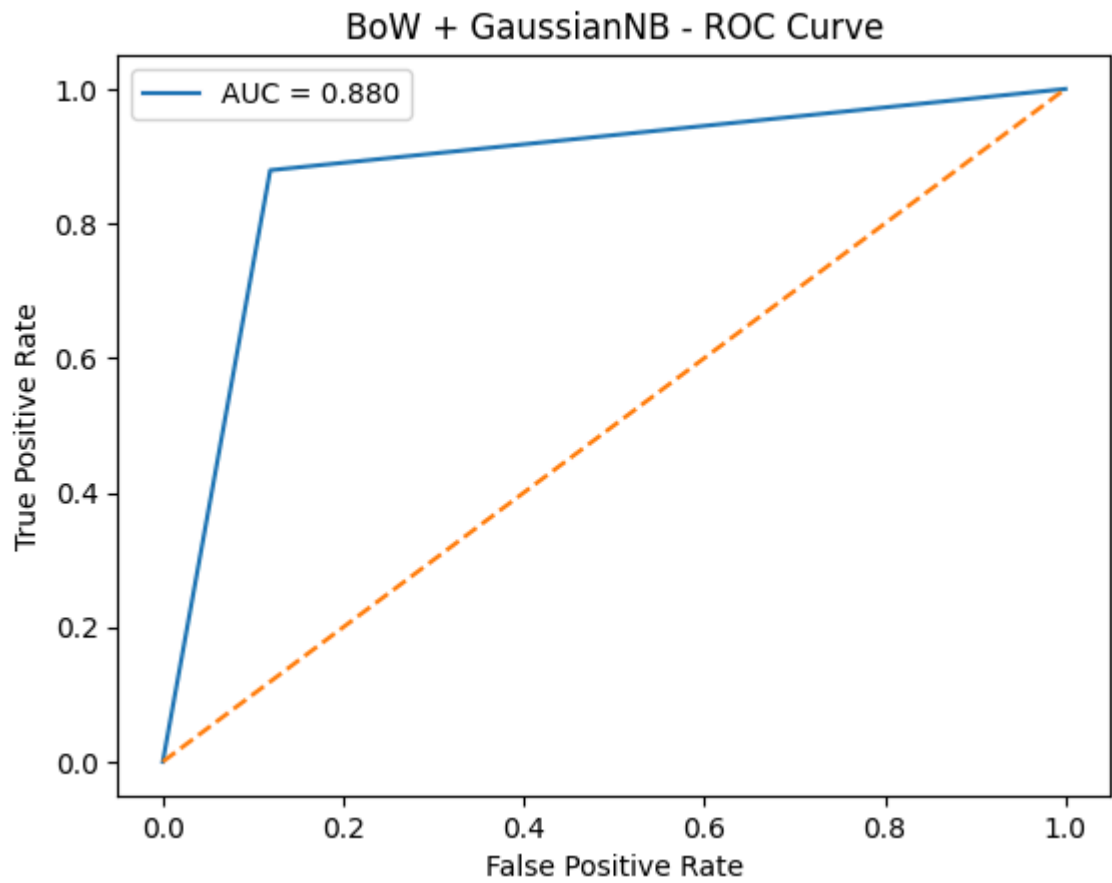
    # Gaussian NB
    gnb = GaussianNB()
    gnb.fit(X_train_vec.toarray(), y_train)
    results.append(
        evaluate_model(
            gnb,
            "gaussian",
            X_test_vec,
            y_test,
            f"{name} + GaussianNB"
        )
    )
```

BoW + MultinomialNB				
	precision	recall	f1-score	support
0	0.98	0.99	0.99	966
1	0.96	0.88	0.92	149
accuracy			0.98	1115
macro avg	0.97	0.94	0.95	1115
weighted avg	0.98	0.98	0.98	1115

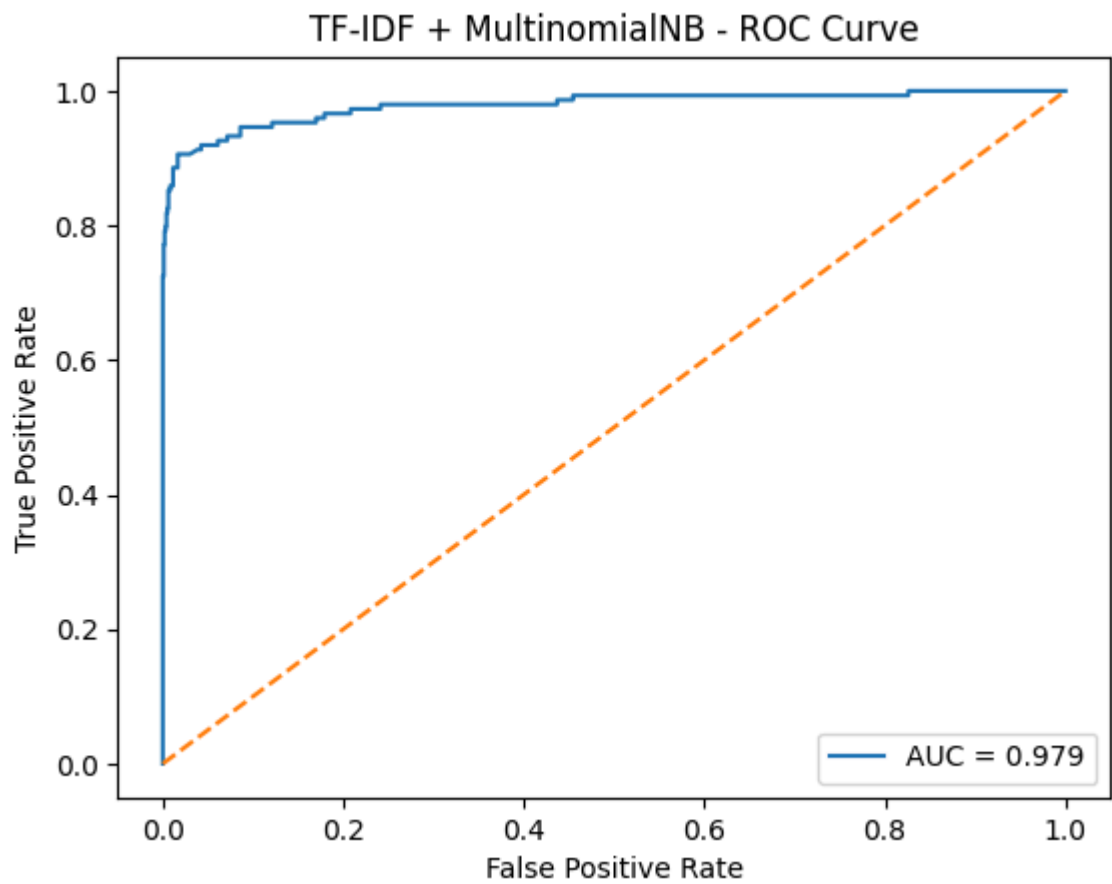
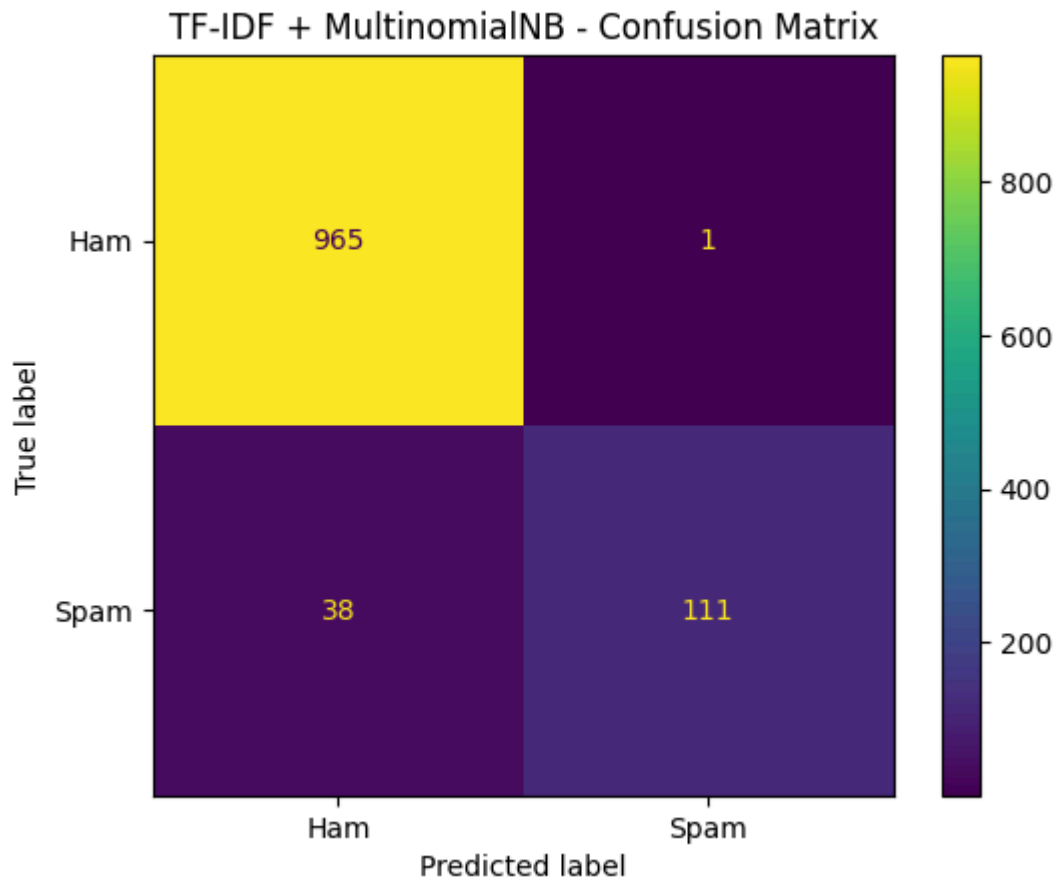


BoW + GaussianNB				
	precision	recall	f1-score	support
0	0.98	0.88	0.93	966
1	0.53	0.88	0.66	149
accuracy			0.88	1115
macro avg	0.76	0.88	0.80	1115
weighted avg	0.92	0.88	0.89	1115

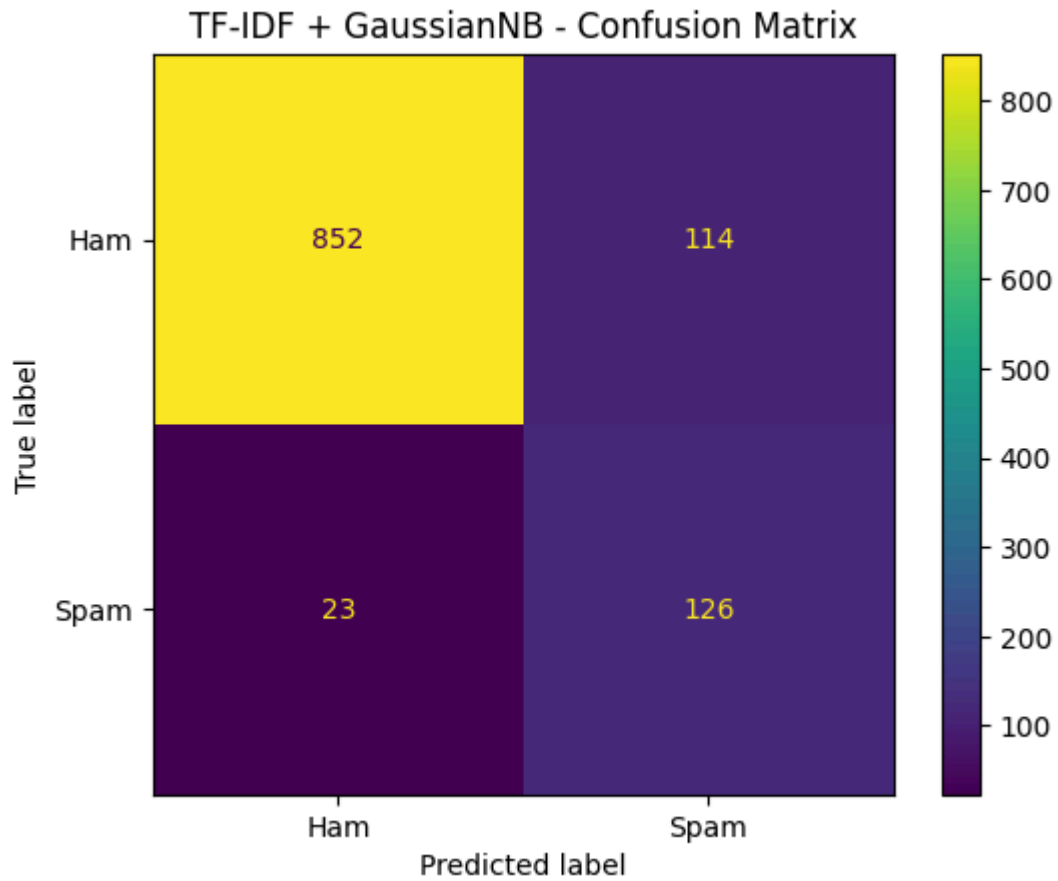


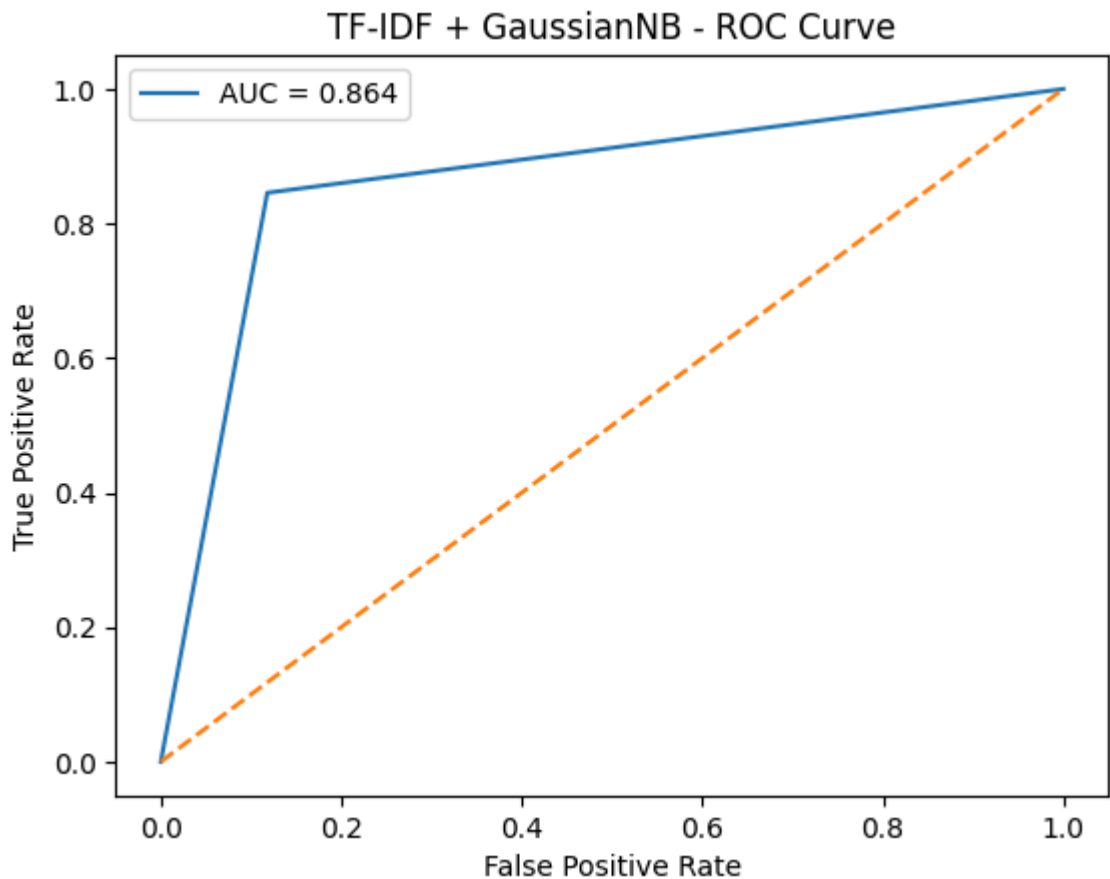


TF-IDF + MultinomialNB					
	precision	recall	f1-score	support	
0	0.96	1.00	0.98	966	
1	0.99	0.74	0.85	149	
accuracy			0.97	1115	
macro avg	0.98	0.87	0.92	1115	
weighted avg	0.97	0.97	0.96	1115	



TF-IDF + GaussianNB					
	precision	recall	f1-score	support	
0	0.97	0.88	0.93	966	
1	0.53	0.85	0.65	149	
accuracy			0.88	1115	
macro avg	0.75	0.86	0.79	1115	
weighted avg	0.91	0.88	0.89	1115	





```
In [32]: results_df = pd.DataFrame(results).sort_values("F1", ascending=False)
results_df
```

```
Out[32]:
```

	Model	Accuracy	Precision	Recall	F1
0	BoW + MultinomialNB	0.978475	0.956204	0.879195	0.916084
2	TF-IDF + MultinomialNB	0.965022	0.991071	0.744966	0.850575
1	BoW + GaussianNB	0.880717	0.532520	0.879195	0.663291
3	TF-IDF + GaussianNB	0.877130	0.525000	0.845638	0.647815

```
In [33]: final_vectorizer = TfidfVectorizer(max_features=5000)
X_all = final_vectorizer.fit_transform(df["clean_text"])

final_model = MultinomialNB()
final_model.fit(X_all, y)

test_emails = [
    "Congratulations you won a free prize",
    "Sir I have submitted the assignment",
    "Urgent your bank account is blocked",
    "Are we meeting tomorrow?"
]

for email in test_emails:
    vec = final_vectorizer.transform([clean_text(email)])
    pred = final_model.predict(vec)[0]
    prob = final_model.predict_proba(vec)[0][1]

    print("\nEmail:", email)
```

```
print("Prediction:", "SPAM" if pred == 1 else "HAM")  
print("Spam Probability:", round(prob, 3))
```

Email: Congratulations you won a free prize
Prediction: SPAM
Spam Probability: 0.98

Email: Sir I have submitted the assignment
Prediction: HAM
Spam Probability: 0.03

Email: Urgent your bank account is blocked
Prediction: SPAM
Spam Probability: 0.514

Email: Are we meeting tomorrow?
Prediction: HAM
Spam Probability: 0.013