Double-click (or enter) to edit

```
import pandas as pd
import re
import string
import pickle
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Load dataset
df = pd.read_csv("/content/combined_data.csv")
```

```
df.head()
```

	label	text
0	1	ounce feather bowl hummingbird opec moment ala...
1	1	wulvob get your medircations online qnb ikud v...
2	0	computer connection from cnn com wednesday es...
3	1	university degree obtain a prosperous future m...
4	0	thanks for all your answers guys i know i shou...

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
# Check for NaN values in 'label'
print("Missing values in label:", df['label'].isna().sum())
```

```
Missing values in label: 0
```

```
# Download NLTK resources
nltk.download('stopwords')
nltk.download('wordnet')

lemmatizer = WordNetLemmatizer()
stop_words = set(stopwords.words("english"))

def clean_email(text):
    text = text.lower()
```

```
    text = re.sub(r'\d+', '', text)  # Remove numbers
    text = text.translate(str.maketrans('', '', string.punctuation))  # Remove punctuation
    text = ' '.join([lemmatizer.lemmatize(word) for word in text.split() if word not in stop_words])  # Lemmatization & Stopword removal
    return text

# Apply text cleaning
df['clean_text'] = df['text'].apply(clean_email)
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
```

```
# Feature extraction
vectorizer = TfidfVectorizer(max_features=5000)
X = vectorizer.fit_transform(df['clean_text'])
y = df['label']
```

```
# Ensure no NaN values in 'y'
y.fillna(y.mode()[0], inplace=True)
```

```
# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train model
model = MultinomialNB()
model.fit(X_train, y_train)
```

```
MultinomialNB
MultinomialNB()
```

```
# Predictions
y_pred = model.predict(X_test)
```

```
# Evaluation
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

```
Accuracy: 0.9651022864019254
Classification Report:
                precision    recall  f1-score   support

              0       0.96      0.96      0.96      1986
              1       0.97      0.97      0.97      2169

       accuracy                           0.97      4155
```

```
           macro avg       0.97      0.96      0.97      4155
        weighted avg       0.97      0.97      0.97      4155

        Confusion Matrix:
         [[1911   75]
         [  70 2099]]
```

```
# Save model and vectorizer
pickle.dump(model, open("email_spam_model.pkl", "wb"))
pickle.dump(vectorizer, open("email_vectorizer.pkl", "wb"))
```

Testing the data

Start coding or generate with AI.

```
def predict_email(email_text):
    email_vectorized = vectorizer.transform([email_text])
    prediction = model.predict(email_vectorized)
    return "Spam" if prediction[0] == 1 else "Not Spam"

# Test with an example email
sample_email = "Congratulations! You've won a free lottery. Claim now."
print("Prediction:", predict_email(sample_email))
```

```
Prediction: Spam
```

```
def predict_email(email_text):
    cleaned_email = clean_email(email_text)  # Clean the text
    email_vectorized = vectorizer.transform([cleaned_email])  # Convert to TF-IDF features
    prediction = model.predict(email_vectorized)  # Predict spam or not
    return "Spam" if prediction[0] == 1 else "Not Spam"

# Test with different emails
test_emails = [
    "Congratulations! You've won $1000! Claim now.",
    "Hello, I hope you're doing well. Let's meet for coffee tomorrow.",
    "Your bank account is compromised! Click here to secure it.",
    "Please find attached the report you requested.",
]

for email in test_emails:
    print(f"Email: {email}")
    print("Prediction:", predict_email(email))
    print("-" * 50)
```

```
Email: Congratulations! You've won $1000! Claim now.
Prediction: Spam
--------------------------------------------------
Email: Hello, I hope you're doing well. Let's meet for coffee tomorrow.
Prediction: Spam
--------------------------------------------------
Email: Your bank account is compromised! Click here to secure it.
Prediction: Spam
--------------------------------------------------
Email: Please find attached the report you requested.
Prediction: Not Spam
--------------------------------------------------
```