```
import pandas as pd
data = pd.read_csv("/content/SpamCollectionSMS.txt", sep='\t', names=["label", "message"])
data.head()
```

	label	message
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...

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

```
#nltk libraries

import nltk
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')

from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

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

# Initialize stemmer and lemmatizer
stemmer = PorterStemmer()
lemmatizer = WordNetLemmatizer()

# Create a list to hold cleaned messages
corpus = []
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

```
# Loop through each message in the dataset
```

```
    for message in data['message']:
        # Remove special characters and numbers
        message = re.sub(r'[^a-zA-Z\s]', '', message)
        message = message.lower()  # Convert to lowercase

        # Tokenize the message
        words = word_tokenize(message)

        # Remove stopwords
        stop_words = set(stopwords.words('english'))
        words = [word for word in words if word not in stop_words]

        # Perform stemming and lemmatization
        words_stemmed = [stemmer.stem(word) for word in words]
        words_lemmatized = [lemmatizer.lemmatize(word) for word in words_stemmed]

        # Join words back into a single string
        cleaned_message = ' '.join(words_lemmatized)
        corpus.append(cleaned_message)

    # Add cleaned messages to the DataFrame
    data['cleaned_message'] = corpus
```

```
    # Split the data
    X = data['cleaned_message']
    y = data['label'].map({'ham': 0, 'spam': 1})  # Convert labels to binary
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=42)
```

```
    # Vectorize the text
    vectorizer = CountVectorizer()
    X_train_vec = vectorizer.fit_transform(X_train)
    X_test_vec = vectorizer.transform(X_test)
```

```
    # Train the model
    spam_detect_model = ComplementNB()
    spam_detect_model.fit(X_train_vec, y_train)
```

```
▾ ComplementNB
ComplementNB()
```

```
    pred = spam_detect_model.predict(X_test_vec)
    pred
```

```
array([0, 1, 0, ..., 0, 0, 0])
```

```
    # Evaluate the model
    confusion_m = confusion_matrix(y_test, pred)
    print("Confusion Matrix:\n", confusion_m)
    print("\nClassification Report:\n", classification_report(y_test, pred))
```

```
Confusion Matrix:
 [[929  36]
 [ 11 139]]

Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.96      0.98       965
           1       0.79      0.93      0.86       150

    accuracy                           0.96      1115
   macro avg       0.89      0.94      0.92      1115
weighted avg       0.96      0.96      0.96      1115
```