lab6

January 20, 2025

[72]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import nltk
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report, confusion_matrix,␣
↪accuracy_score
```

[79]:
```
df = pd.read_csv("./1.ushape.csv", header= None, names=["label","text",␣
↪"class"])
```

[ ]:
```
stemmer = PorterStemmer()

def clean_text(message):
    message = message.lower()

    tokens = nltk.word_tokenize(message)

    a = [t for t in a if t.isalpha() and t not in q]

    tokens = [stemmer.stem(word) for word in tokens]

    return " ".join(tokens)
```

[47]:
```
df['cleaned_text'] = df['text'].apply(clean_text)
df[['text', 'cleaned_text']].head()
```

[47]:
```
Empty DataFrame
Columns: [text, cleaned_text]
Index: []
```

[51]:
```
def clean_text(message):
    # Handle non-string inputs
    if not isinstance(message, str):
        print("Invalid input:", message)
        return ""

    # Lowercase
    message = message.lower()
    print("Lowercased:", message)

    # Tokenize
    tokens = nltk.word_tokenize(message)
    print("Tokens:", tokens)

    # Remove non-alphabetic words and stopwords
    tokens = [word for word in tokens if word.isalpha() and word not in␣
 ↪stop_words]
    print("After removing stopwords:", tokens)

    # Stem words
    tokens = [stemmer.stem(word) for word in tokens]
    print("After stemming:", tokens)

    return " ".join(tokens)

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

[53]:
```
# Convert all values in the 'text' column to strings
df['text'] = df['text'].astype(str)
df['text'] = df['text'].fillna("")

# Remove rows with completely empty text
df = df[df['text'].str.strip() != ""]
```

[54]:
```
print("Sample data from 'text' column:")
print(df['text'].head())

# Check for empty or NaN values
print("Number of NaN values in 'text':", df['text'].isna().sum())
print("Number of completely empty rows:", (df['text'].str.strip() == "").sum())
```

```
Sample data from 'text' column:
Series([], Name: text, dtype: object)
Number of NaN values in 'text': 0
Number of completely empty rows: 0
```

```
[59]: y = df['label'].map({'ham': 0, 'spam': 1})
```

```
[62]: y
```

```
[62]: Series([], Name: label, dtype: int64)
```