

```

import string
import nltk
import numpy as np
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer,
TfidfTransformer
from sklearn.naive_bayes import MultinomialNB

# Download necessary NLTK data
nltk.download('stopwords')

[nltk_data] Downloading package stopwords to
[nltk_data] C:\Users\Asus\AppData\Roaming\nltk_data...
[nltk_data] Package stopwords is already up-to-date!

True

# Define text preprocessing function
def textPreprocessing(data):
    if not isinstance(data, str):
        return ""
    remove_pun = [c for c in data if c not in string.punctuation]
    sentences = ''.join(remove_pun)
    words = sentences.split()
    return ' '.join(words)

# Load dataset
file_path = r"spam.csv" # Use raw string to handle backslashes in the
file path
df = pd.read_csv(file_path, sep='\t', names=['label', 'message'],
encoding='latin1')
df['message'] = df['message'].astype(str)

wordVector = CountVectorizer(analyzer=textPreprocessing)
finalWordVector = wordVector.fit(df['message'])
print(finalWordVector.vocabulary_)
bow = finalWordVector.transform(df['message'])

print(bow)

{'n': 1, 'a': 0}
(0, 0) 1
(0, 1) 2
(1, 0) 1
(1, 1) 2
(2, 0) 1
(2, 1) 2
(3, 0) 1
(3, 1) 2
(4, 0) 1

```

```

(4, 1) 2
(5, 0) 1
(5, 1) 2
(6, 0) 1
(6, 1) 2
(7, 0) 1
(7, 1) 2
(8, 0) 1
(8, 1) 2
(9, 0) 1
(9, 1) 2
(10, 0) 1
(10, 1) 2
(11, 0) 1
(11, 1) 2
(12, 0) 1
:
:
(5562, 1) 2
(5563, 0) 1
(5563, 1) 2
(5564, 0) 1
(5564, 1) 2
(5565, 0) 1
(5565, 1) 2
(5566, 0) 1
(5566, 1) 2
(5567, 0) 1
(5567, 1) 2
(5568, 0) 1
(5568, 1) 2
(5569, 0) 1
(5569, 1) 2
(5570, 0) 1
(5570, 1) 2
(5571, 0) 1
(5571, 1) 2
(5572, 0) 1
(5572, 1) 2
(5573, 0) 1
(5573, 1) 2
(5574, 0) 1
(5574, 1) 2

```

Transform to TF-IDF features

```
tfidfObject = TfidfTransformer().fit(bow)
final_feature = tfidfObject.transform(bow)
```

Train the Naive Bayes model

```
model = MultinomialNB()
model.fit(final_feature, df['label'])
```

```
MultinomialNB()

# Evaluate the model
score = model.score(final_feature, df['label'])
print("Model Accuracy: ", score)

Model Accuracy:  0.0053811659192825115

# Input SMS for prediction
inputSMS = input("Enter the SMS Content: ")
preprocessText = textPreprocessing(inputSMS)

Enter the SMS Content: random

# Transform the input SMS to feature vector
vector = finalWordVector.transform([preprocessText])
finalFeature = tfidfObject.transform(vector)

# Predict and print the result
pred = model.predict(finalFeature)[0]
print("Given SMS is", pred)

Given SMS is ham,"Sorry, I'll call later",,,
```