```
# Appendix C16 - test_feature_extractor.py

from klassify.src.feature_extractor import FeatureExtractor
from klassify.src.tables import Document

initial_document_1 = Document(title="Test title 1",
                              base_path="/test-1",
                              content="This is a test document - one")
initial_document_2 = Document(title="Test title 2",
                               base_path="/test-2",
                               content="This is a test document - two")
initial_document_3 = Document(title="Test title 3",
                               base_path="/test-3",
                               content="This is a test document - three")

EXTRACTOR = FeatureExtractor([
    initial_document_1,
    initial_document_2,
    initial_document_3,
])

new_document = Document(title="Self assessment deadlines 3",
                        base_path="/self-assessment-3",
                        html="<strong>PAY NOW 3</strong>",
                        content="This has a different content - four")

def test_tokenize():
    tokenized_content = EXTRACTOR.tokenize(initial_document_1)
    assert tokenized_content == ['This', 'is', 'a', 'test', 'document', "-", 'one']

def test_make_vocabulary():
    # without document
    assert EXTRACTOR.make_vocabulary() == ['test', 'one', 'test', 'two', 'test', 'three']
    # with document
    assert EXTRACTOR.make_vocabulary(new_document) ==  ['differ', 'content', 'four']

def test_bag_of_words():
    # This is built against the vocabulary.
    # The vocabulary is the sum of all the different terms in all the documents provided
at instantiation.
    assert EXTRACTOR.bag_of_words(initial_document_3) == {'one': False, 'test': True, 'th
ree': True, 'two': False}
    assert EXTRACTOR.bag_of_words(new_document) == {'one': False, 'test': False, 'three':
 False, 'two': False}

def test_process():
    # What is bein discarded: Single letter words, Stop words, Long words
    # Additionally, remaining words will be stemmed.
    document_with_unfiltered_content = Document(title="Test", base_path="/test",
        content=" within https .mb , a b c reallylongwordthatshouldbefilteredout cloudy r
egular words should be stemmed in this process"
    )

    tokenized_content = EXTRACTOR.tokenize(document_with_unfiltered_content)

    assert EXTRACTOR.process(tokenized_content) == ['cloudi', 'regular', 'word', 'stem',
'process']

```