```python
import nltk
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from gensim.models import Word2Vec, FastText
from sklearn.preprocessing import normalize
from nltk.tokenize import sent_tokenize, word_tokenize
from nltk.corpus import stopwords
import string

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

```
[nltk_data] Downloading package punkt to /home/ai-ds2/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package stopwords to /home/ai-ds2/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

```
True
```

```python
# Example input text
document = """
Machine learning is a branch of artificial intelligence that focuses
on building systems that can learn from data.
It enables machines to make decisions without being explicitly
programmed.
Machine learning has various applications, such as image recognition,
natural language processing, and self-driving cars.
It has also led to breakthroughs in areas like healthcare and finance.

However, machine learning models can sometimes be biased or make
incorrect predictions.
Ensuring fairness in machine learning models is an ongoing challenge.
"""

# Preprocessing function
def preprocess_text(text):
    # Lowercasing
    text = text.lower()
    # Removing punctuation
    text = text.translate(str.maketrans('', '', string.punctuation))
    # Tokenization
    tokens = word_tokenize(text)
    # Removing stopwords
    stop_words = set(stopwords.words('english'))
    tokens = [word for word in tokens if word not in stop_words]
    return ' '.join(tokens)
```

```
# Preprocess the input document
document = preprocess_text(document)
sentences = sent_tokenize(document)

# Sentence Embedding Techniques

## 1. Bag of Words (BoW)
def get_bow_vector(sentences):
    vectorizer = TfidfVectorizer()
    vectors = vectorizer.fit_transform(sentences)
    return vectors

## 2. TF-IDF
def get_tfidf_vector(sentences):
    vectorizer = TfidfVectorizer()
    vectors = vectorizer.fit_transform(sentences)
    return vectors

## 3. Word2Vec
def get_word2vec_vectors(sentences):
    # Tokenizing sentences into words
    tokenized_sentences = [sentence.split() for sentence in sentences]
    # Training Word2Vec model
    model = Word2Vec(tokenized_sentences, min_count=1)
    # Sentence vectors: average of word vectors
    sentence_vectors = []
    for sentence in tokenized_sentences:
        vectors = [model.wv[word] for word in sentence if word in
model.wv]
        if vectors:
            sentence_vectors.append(np.mean(vectors, axis=0))
        else:
            sentence_vectors.append(np.zeros(model.vector_size))
    return sentence_vectors

## 4. FastText
def get_fasttext_vectors(sentences):
    # Tokenizing sentences into words
    tokenized_sentences = [sentence.split() for sentence in sentences]
    # Training FastText model
    model = FastText(tokenized_sentences, min_count=1)
    # Sentence vectors: average of word vectors
    sentence_vectors = []
    for sentence in tokenized_sentences:
        vectors = [model.wv[word] for word in sentence if word in
model.wv]
        if vectors:
            sentence_vectors.append(np.mean(vectors, axis=0))
        else:
```

```
            sentence_vectors.append(np.zeros(model.vector_size))
    return sentence_vectors

# Calculate sentence similarity using cosine similarity

def calculate_similarity(sentence_vectors):
    # Normalizing the vectors
    sentence_vectors = normalize(sentence_vectors)
    similarity_matrix = cosine_similarity(sentence_vectors)
    return similarity_matrix

# Ranking sentences based on their importance (sum of similarity
scores)

def extract_summary(similarity_matrix, sentences,
num_summary_sentences=3):
    scores = similarity_matrix.sum(axis=1)  # Sum of similarities for
each sentence
    ranked_sentences_idx = np.argsort(scores)[::-1]  # Sort sentences
based on scores
    summary = [sentences[i] for i in
ranked_sentences_idx[:num_summary_sentences]]
    return summary

# Generating sentence vectors for each method

## 1. BoW Vector
bow_vectors = get_bow_vector(sentences)

## 2. TF-IDF Vector
tfidf_vectors = get_tfidf_vector(sentences)

## 3. Word2Vec Vectors
word2vec_vectors = get_word2vec_vectors(sentences)

## 4. FastText Vectors
fasttext_vectors = get_fasttext_vectors(sentences)

# Compute sentence similarity for each method

bow_similarity = calculate_similarity(bow_vectors.toarray())
tfidf_similarity = calculate_similarity(tfidf_vectors.toarray())
word2vec_similarity = calculate_similarity(word2vec_vectors)
fasttext_similarity = calculate_similarity(fasttext_vectors)

# Extracting summaries using similarity matrices

bow_summary = extract_summary(bow_similarity, sentences)
tfidf_summary = extract_summary(tfidf_similarity, sentences)
word2vec_summary = extract_summary(word2vec_similarity, sentences)
fasttext_summary = extract_summary(fasttext_similarity, sentences)
```

```python
# Display the summaries

print("Extractive Summary based on BoW:")
print(bow_summary)
```

```
Extractive Summary based on BoW:
['machine learning branch artificial intelligence focuses building
systems learn data enables machines make decisions without explicitly
programmed machine learning various applications image recognition
natural language processing selfdriving cars also led breakthroughs
areas like healthcare finance however machine learning models
sometimes biased make incorrect predictions ensuring fairness machine
learning models ongoing challenge']
```

```python
print("\nExtractive Summary based on TF-IDF:")
print(tfidf_summary)
```

```

Extractive Summary based on TF-IDF:
['machine learning branch artificial intelligence focuses building
systems learn data enables machines make decisions without explicitly
programmed machine learning various applications image recognition
natural language processing selfdriving cars also led breakthroughs
areas like healthcare finance however machine learning models
sometimes biased make incorrect predictions ensuring fairness machine
learning models ongoing challenge']
```

```python
print("\nExtractive Summary based on Word2Vec:")
print(word2vec_summary)
```

```

Extractive Summary based on Word2Vec:
['machine learning branch artificial intelligence focuses building
systems learn data enables machines make decisions without explicitly
programmed machine learning various applications image recognition
natural language processing selfdriving cars also led breakthroughs
areas like healthcare finance however machine learning models
sometimes biased make incorrect predictions ensuring fairness machine
learning models ongoing challenge']
```

```python
print("\nExtractive Summary based on FastText:")
print(fasttext_summary)
```

```

Extractive Summary based on FastText:
['machine learning branch artificial intelligence focuses building
systems learn data enables machines make decisions without explicitly
programmed machine learning various applications image recognition
natural language processing selfdriving cars also led breakthroughs
areas like healthcare finance however machine learning models
```

```
sometimes biased make incorrect predictions ensuring fairness machine
learning models ongoing challenge']
```