NumPy Universal Functions (ufuncs)

✅ **What are ufuncs?**

- **Ufunc** stands for **Universal Function**.
- A **ufunc** performs **fast, element-wise operations** on NumPy arrays.
- They are highly optimized **vectorized operations**, so you don't need explicit Python loops.
- Example: np.add, np.subtract, np.multiply, np.divide are all ufuncs.

✅ **Why use ufuncs?**

- Faster than Python loops (uses compiled C code under the hood).
- Supports **broadcasting**: arrays of different shapes work together.
- Handles **multi-dimensional data** efficiently.
- Reduces your code to clean, single-line operations.

✅ **Most Common ufuncs**

Here's a table of popular ufuncs and what they do:

Function	Purpose	Example
np.add(x, y)	Addition	np.add([1,2], [3,4]) → [4,6]
np.subtract(x, y)	Subtraction	np.subtract([5,6], [1,2]) → [4,4]
np.multiply(x, y)	Multiplication	np.multiply([2,3], [4,5]) → [8,15]
np.divide(x, y)	Division	np.divide([8,9], [2,3]) → [4,3]
np.power(x, y)	Exponentiation	np.power([2,3], 2) → [4,9]
np.exp(x)	Exponential	np.exp([0,1]) → [1, 2.718]
np.sqrt(x)	Square root	np.sqrt([4,9]) → [2,3]
np.log(x)	Natural log	np.log([1, np.e]) → [0,1]
np.sin(x)	Sine	np.sin([0, np.pi/2]) → [0,1]
np.maximum(x, y)	Max element-wise	np.maximum([1,2], [2,1]) → [2,2]

✅ **Broadcasting with ufuncs**

One of the most powerful parts of ufuncs is **broadcasting**.

Example:

python

CopyEdit

```
import numpy as np

a = np.array([1, 2, 3])

b = 2

result = np.add(a, b)  # [3, 4, 5]
```

Here, b is a scalar — NumPy broadcasts it to [2,2,2].

✅ **Vectorization vs. Loops**

Python loop:

python

CopyEdit

```
result = []

for x, y in zip([1, 2, 3], [4, 5, 6]):

   result.append(x + y)
```

Same with ufunc:

python

CopyEdit

```
np.add([1, 2, 3], [4, 5, 6])
```

Much faster and cleaner!

✅ **Advanced: out and where**

Most ufuncs accept:

- out → save result in an existing array (memory-efficient).
- where → apply conditionally.

Example:

python

CopyEdit

```
import numpy as np
```

```
x = np.array([1, 2, 3])
y = np.array([4, 5, 6])
out = np.empty(3)

np.add(x, y, out=out, where=[True, False, True])
print(out)  # [5, any random value, 9]
```

✅ **Creating your own ufunc**

For custom element-wise ops:

python

CopyEdit

```
vectorized_func = np.frompyfunc(lambda x: x + 2, 1, 1)
result = vectorized_func([1, 2, 3])  # [3, 4, 5]
```

🎉 **Key Takeaways**

- Ufuncs are **super fast**, **element-wise**, and **broadcast-ready**.
- They work on **scalars, vectors, matrices**.
- Always prefer ufuncs over explicit loops in NumPy!

Stop Words in NLP + Stemming + Lemmatization

What are Stop Words?

Stop words are common words in a language such as 'the', 'is', 'in', 'and', etc., which are usually removed during text preprocessing because they add little semantic value.

Why Remove Stop Words?

- To reduce noise in text data.

- To decrease dimensionality of feature space.

- To improve computational efficiency.

1. Using NLTK Stop Words

NLTK provides a corpus of stop words. Key functions/methods:

- stopwords.words(): Get the list of stop words for a language.

- word_tokenize(): Tokenize a sentence.

```
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize


text = "This is an example showing off stop word filtration."


stop_words = set(stopwords.words('english'))
words = word_tokenize(text)


filtered = [w for w in words if w.lower() not in stop_words]


print(filtered)  # ['example', 'showing', 'stop', 'word', 'filtration', '.']
```

2. Using spaCy Stop Words

spaCy provides a built-in list of stop words. Key attributes/functions:

- nlp.Defaults.stop_words: Access the stop word list.

- Token.is_stop: Check if a token is a stop word.

- Customize: Add or remove stop words.

```
import spacy
```

```
nlp = spacy.load("en_core_web_sm")
doc = nlp("This is an example showing off stop words.")

for token in doc:
    print(token.text, token.is_stop)

# Example: Add 'showing' to stop words
nlp.Defaults.stop_words.add("showing")
```

3. Using Scikit-learn Stop Words

Scikit-learn provides a built-in list for vectorizers.

- ENGLISH_STOP_WORDS: A frozenset of stop words.

- Can be passed to CountVectorizer or TfidfVectorizer.

```
from sklearn.feature_extraction.text import ENGLISH_STOP_WORDS
from sklearn.feature_extraction.text import CountVectorizer

print(list(ENGLISH_STOP_WORDS)[:10])

vectorizer = CountVectorizer(stop_words='english')
docs = ["This is an example.", "We remove the stop words."]
X = vectorizer.fit_transform(docs)

print(vectorizer.get_feature_names_out())  # ['example' 'remove' 'stop' 'words']
```

4. Custom Stop Words

You can define your own stop word list for special use cases.

Example:

```
custom_stop_words = {'this', 'is', 'an', 'the'}
text = "This is an example of custom stop words."
words = text.lower().split()
filtered = [w for w in words if w not in custom_stop_words]

print(filtered)  # ['example', 'of', 'custom', 'stop', 'words.']
```

5. Stop Words Removal + Stemming

Example program that removes stop words and then applies stemming using NLTK's PorterStemmer:

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

text = "This is an example showing how to filter stop words and apply stemming."

stop_words = set(stopwords.words('english'))
words = word_tokenize(text)

# Remove stop words
filtered = [w for w in words if w.lower() not in stop_words]

# Apply stemming
ps = PorterStemmer()
stemmed = [ps.stem(w) for w in filtered]

print("Filtered:", filtered)
print("Stemmed:", stemmed)
```

6. Stop Words Removal + Lemmatization

Example program that removes stop words and then applies lemmatization using NLTK's WordNetLemmatizer:

```
from nltk.corpus import stopwords, wordnet
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer

text = "The children are playing happily while the dogs are barking."

stop_words = set(stopwords.words('english'))
words = word_tokenize(text)
```

```
# Remove stop words
filtered = [w for w in words if w.lower() not in stop_words]

# Apply lemmatization
lemmatizer = WordNetLemmatizer()
lemmatized = [lemmatizer.lemmatize(w, pos='v') for w in filtered]

print("Filtered:", filtered)
print("Lemmatized:", lemmatized)
```