CS-37 Machine Learning with Python

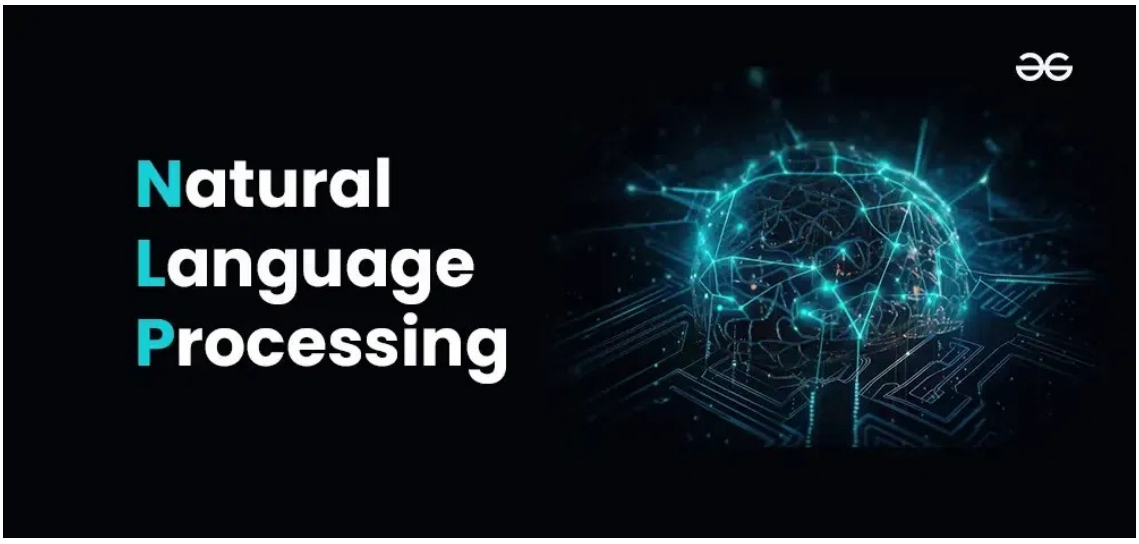
Unit– 04 Natural Language Processing

- Natural Language Processing:
- pre-processing data
- stemming data
- using lemmatization
- diving chunks
- text classifier case study implementation using Python.

Natural Language Processing (NLP)

What is NLP?

- ✓ NLP stands for Natural Language Processing, which is a part of Computer Science, Human language, and Artificial Intelligence.
- ✓ It is the technology that is used by machines to understand, analyse, manipulate, and interpret human's languages.
- ✓ It helps developers to organize knowledge for performing tasks such as translation, automatic summarization, Named Entity Recognition (NER), speech recognition, relationship extraction, and topic segmentation.



Here's an easy-to-learn, step-by-step explanation of Natural Language Processing (NLP) concepts:

1. Breaking Down Text

- **Tokenization** → Splitting text into words or phrases (tokens).
- **Stopword Removal** → Removing common words like "the," "is," "and" to focus on important words.

2. Understanding Words

- **Stemming** → Cutting words to their root form (e.g., "running" → "run").
- **Lemmatization** → Converting words to their dictionary form (e.g., "better" → "good").

3. Understanding Sentence Structure

- **Part-of-Speech (POS) Tagging** → Identifying nouns, verbs, adjectives, etc.
- **Parsing** → Analyzing sentence grammar and structure.

4. Identifying Important Information

- **Named Entity Recognition (NER)** → Finding names, places, dates in text.
- **Relation Extraction** → Understanding relationships between words (e.g., "Apple acquired Beats").

5. Understanding Meaning and Emotions

- **Sentiment Analysis** → Detecting positive, negative, or neutral emotions in text.

- **Topic Modeling** → Finding key topics in large text collections.

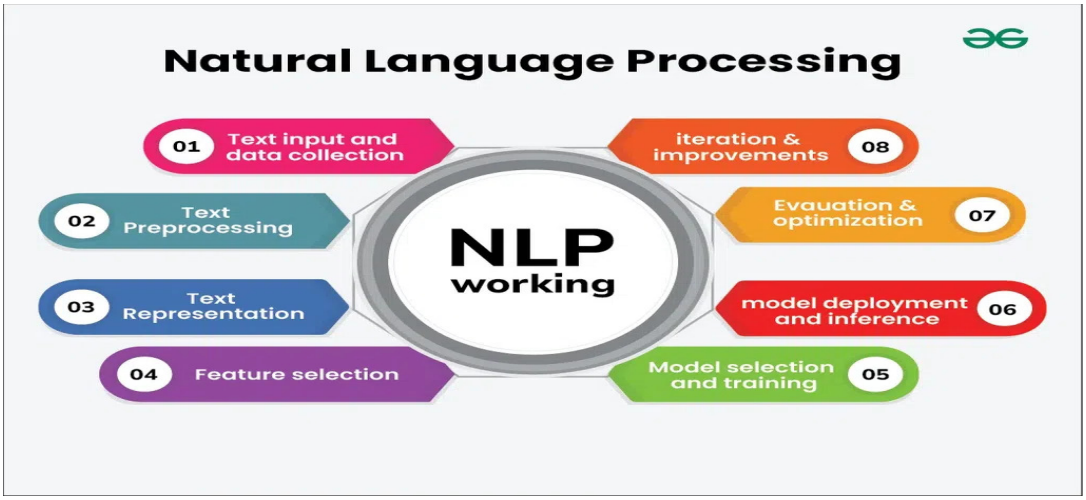
6. Advanced Applications

- **Machine Translation** → Converting text from one language to another (e.g., English to Spanish).
- **Text Summarization** → Creating short summaries from large documents.
- **Chatbots & Virtual Assistants** → AI-powered conversations (e.g., Siri, Alexa).

7. Speech and Question Processing

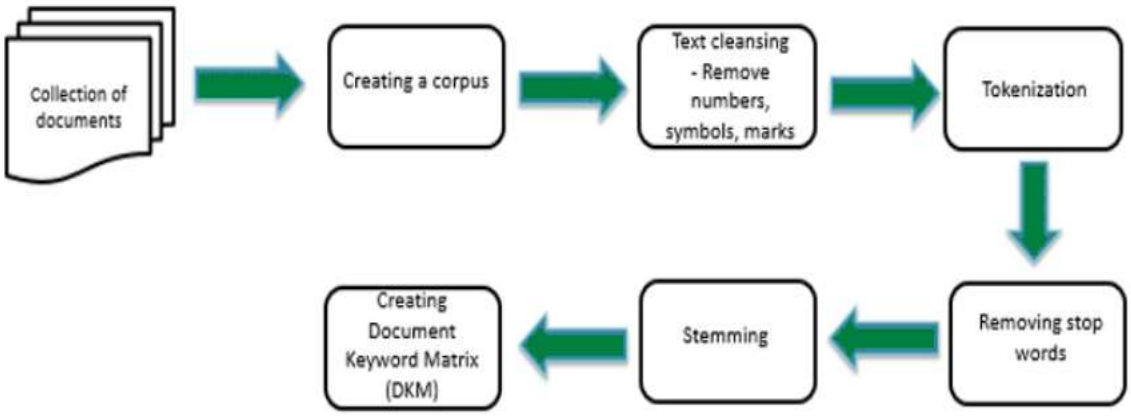
- **Speech Recognition** → Converting spoken words into text.
- **Question Answering (QA)** → Extracting precise answers from text (e.g., Google search).

Working of Natural Language Processing (NLP)



Text Preprocessing in NLP

- ✓ Natural Language Processing (NLP) has seen tremendous growth and development, becoming an integral part of various applications, from chatbots to sentiment analysis.
- ✓ One of the foundational steps in NLP is text preprocessing, which involves cleaning and preparing raw text data for further analysis or model training.
- ✓ Proper text preprocessing can significantly impact the performance and accuracy of NLP models.
- ✓ This article will delve into the essential steps involved in text preprocessing for NLP tasks.



Why Text Preprocessing is Important?

Raw text data is often noisy and unstructured, containing various inconsistencies such as typos, slang, abbreviations, and irrelevant information. Preprocessing helps in:

- **Improving Data Quality:** Removing noise and irrelevant information ensures that the data fed into the model is clean and consistent.
- **Enhancing Model Performance:** Well-preprocessed text can lead to better feature extraction, improving the performance of NLP models.
- **Reducing Complexity:** Simplifying the text data can reduce the computational complexity and make the models more efficient.

Text Preprocessing Technique in NLP

1. Regular Expressions:

Regular expressions (regex) are a powerful tool in text preprocessing for Natural Language Processing (NLP). They allow for efficient and flexible pattern matching and text manipulation.

2. Tokenization:

Tokenization is the process of breaking down text into smaller units, such as words or sentences. This is a crucial step in NLP as it transforms raw text into a structured format that can be further analyzed.

3. Lemmatization and Stemming:

Lemmatization and stemming are techniques used in NLP to reduce words to their base or root forms. This process is

important for tasks like text normalization, information retrieval, and text mining.

4. Parts of Speech (POS):

Parts of Speech (POS) tagging is a fundamental task in NLP that involves labeling each word in a sentence with its corresponding part of speech, such as noun, verb, adjective, etc. This information is crucial for many NLP applications, including parsing, information retrieval, and text analysis.

Introduction to Stemming

- ✓ **Stemming** is a method in **text processing** that eliminates prefixes and suffixes from words, transforming them into their fundamental or root form, The main objective of stemming is to streamline and standardize words, enhancing the effectiveness of the **natural language processing** tasks.

- ✓ For example, “chocolates” becomes “chocolate” and “retrieval” becomes “retrieve.” This is crucial for pipelines for natural language processing, which use

tokenized words that are acquired from the first stage of dissecting a document into its constituent words.

Why is Stemming important?

✓ It is important to note that stemming is different from Lemmatization. Lemmatization is the process of reducing a word to its base form, but unlike stemming, it takes into account the context of the word, and it produces a valid word, unlike stemming which may produce a non-word as the root form.

Some more example of stemming for root word "like" include:
->"likes"
->"liked"
->"likely"
->"liking"

Types of Stemmer in NLP

1. Porter's Stemmer
2. Lovins Stemmer
3. Dawson Stemmer
4. Snowball Stemmer
5. Lancaster Stemmer

Example: EED -> EE means "if the word has at least one vowel and consonant plus EED ending, change the ending to EE" as 'agreed' becomes 'agree'.

Implementation of Porter Stemmer(Install nltk Module)

```
from nltk.stem import PorterStemmer

# Create a Porter Stemmer instance
porter_stemmer = PorterStemmer()

# Example words for stemming
words = ["running", "jumps", "happily", "running",
```

```
"happily"]

# Apply stemming to each word
stemmed_words = [porter_stemmer.stem(word) for word in
words]

# Print the results
print("Original words:", words)
print("Stemmed words:", stemmed_words)
```

Output:

```
Original words: ['running', 'jumps', 'happily', 'running',
'happily']
Stemmed words: ['run', 'jump', 'happili', 'run', 'happili']
```

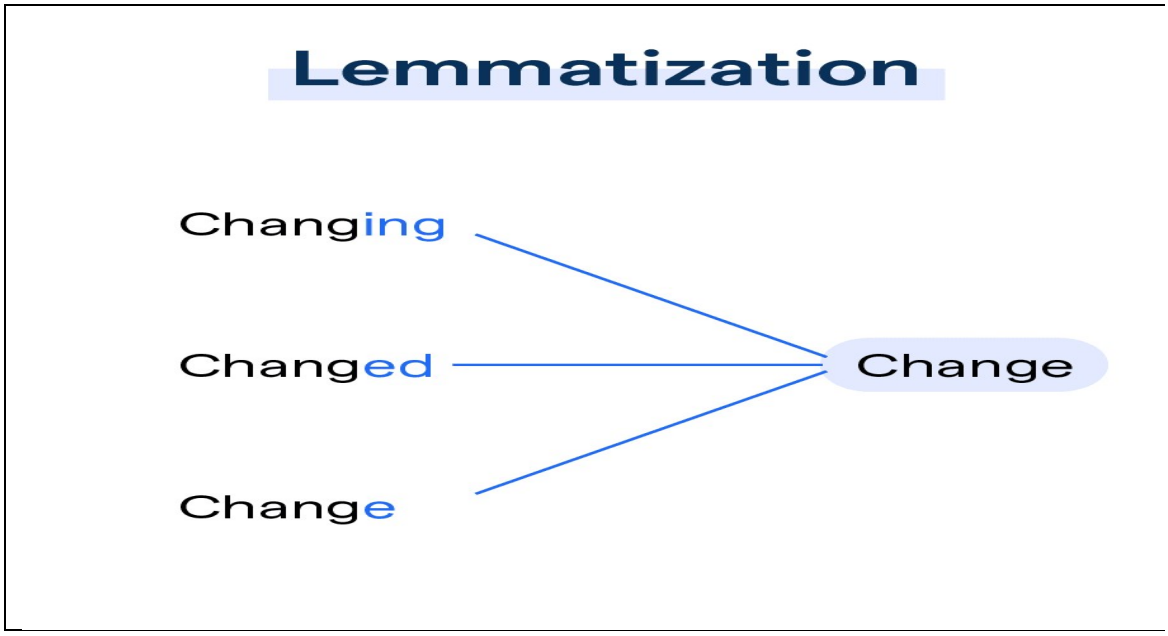
Applications of Stemming

1. Stemming is used in information retrieval systems like search engines.
2. It is used to determine domain vocabularies in domain analysis.
3. To display search results by indexing while documents are evolving into numbers and to map documents to common subjects by stemming.

Lemmatization

What is Lemmatization?

- ✓ Lemmatization is the process of reducing words to their base or dictionary form, known as the lemma.
- ✓ This technique considers the context and the meaning of the words, ensuring that the base form belongs to the language's dictionary.
- ✓ For example, the words "running," "ran," and "runs" are all lemmatized to the lemma "run."



How Lemmatization Works?

Lemmatization involves several steps:

1. **Part-of-Speech (POS) Tagging**: Identifying the grammatical category of each word (e.g., noun, verb, adjective).

2. **Morphological Analysis**: Analyzing the structure of the word to understand its root form.
3. **Dictionary Lookup**: Using a predefined vocabulary to find the lemma of the word.

For example, the word "better" would be lemmatized to "good" if it is identified as an adjective, whereas "running" would be lemmatized to "run" if identified as a verb.

Real-World Application Of Lemmatization

Techniques in Lemmatization

1. **Rule-Based Lemmatization**: Uses predefined grammatical rules to transform words. For instance, removing the "-ed" suffix from regular past tense verbs.
2. **Dictionary-Based Lemmatization**: Looks up words in a dictionary to find their base forms.
3. **Machine Learning-Based Lemmatization**: Employs machine learning models trained on annotated corpora to predict the lemma of a word.

Advantages and Disadvantages of Lemmatization

Benefits:

- **Accuracy:** Lemmatization provides more accurate results because it considers the context and meaning of words.
- **Standardization:** Ensures words are reduced to their dictionary form, aiding in tasks like text normalization and information retrieval.

Limitations:

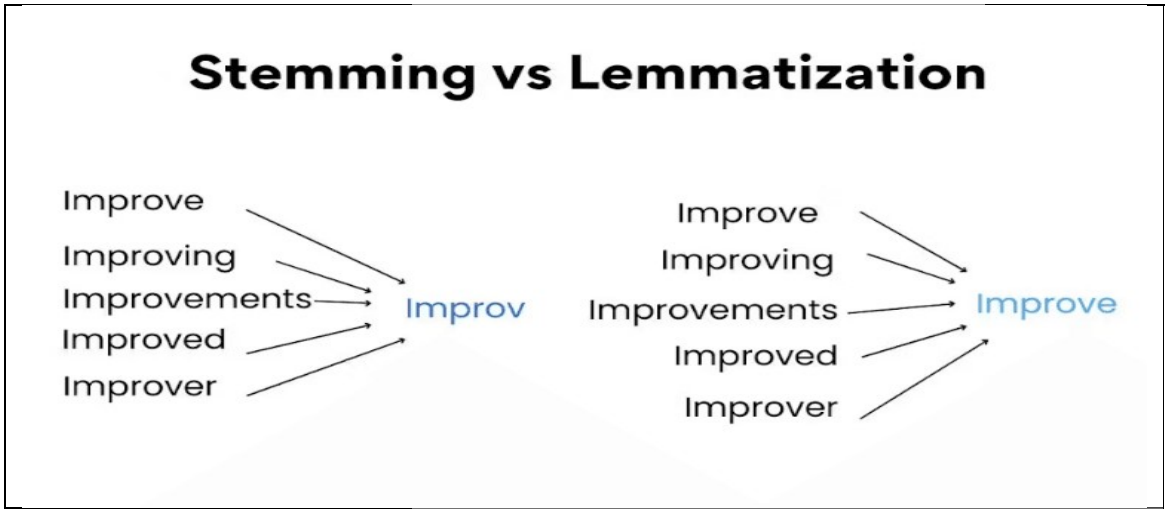
- **Complexity:** Requires more computational resources and a comprehensive dictionary.
- **Dependency on POS Tagging:** Requires accurate POS tagging, which adds to the processing overhead.

Lemmatization vs. Stemming: Key Differences

Aspect	Lemmatization	Stemming
Definition	Converts words to their base or dictionary form (lemma).	Reduces words to their root form (stem), which may not be a valid word.

Aspect	Lemmatization	Stemming
Complexity	Higher complexity, context-aware.	Lower complexity, context-agnostic.
Algorithms	Uses dictionaries and morphological analysis.	Uses rule-based algorithms like Porter, Snowball, and Lancaster Stemmers.
Accuracy	Produces more accurate and meaningful words.	Less accurate, may produce non-meaningful stems.
Output Example	"Running" → "run", "Better" → "good".	"Running" → "run" or "runn", "Better" → "bett".
Speed	Slower due to more complex processing.	Faster due to simpler rules.
Use in Search Engines	Better search results through understanding context.	Useful for quick search indexing
Text Analysis	Essential for tasks needing accurate word forms (e.g.,	Used for initial stages of preprocessing to

Aspect	Lemmatization	Stemming
	sentiment analysis, topic modeling)	reduce word variability
Machine Translation	Helps in producing grammatically correct translations	Less common due to potential inaccuracy
Information Retrieval	Suitable for detailed and precise analysis	Useful for reducing data dimensionality



Diving Chunks

- ✓ "Chunking" in machine learning refers to the process of dividing large pieces of data, particularly text, into smaller, more manageable segments called "chunks," which allows models to process information

more efficiently, especially when dealing with lengthy documents or complex datasets; essentially, it's like breaking down a large book into smaller chapters to better understand its content.

Key points about chunking:

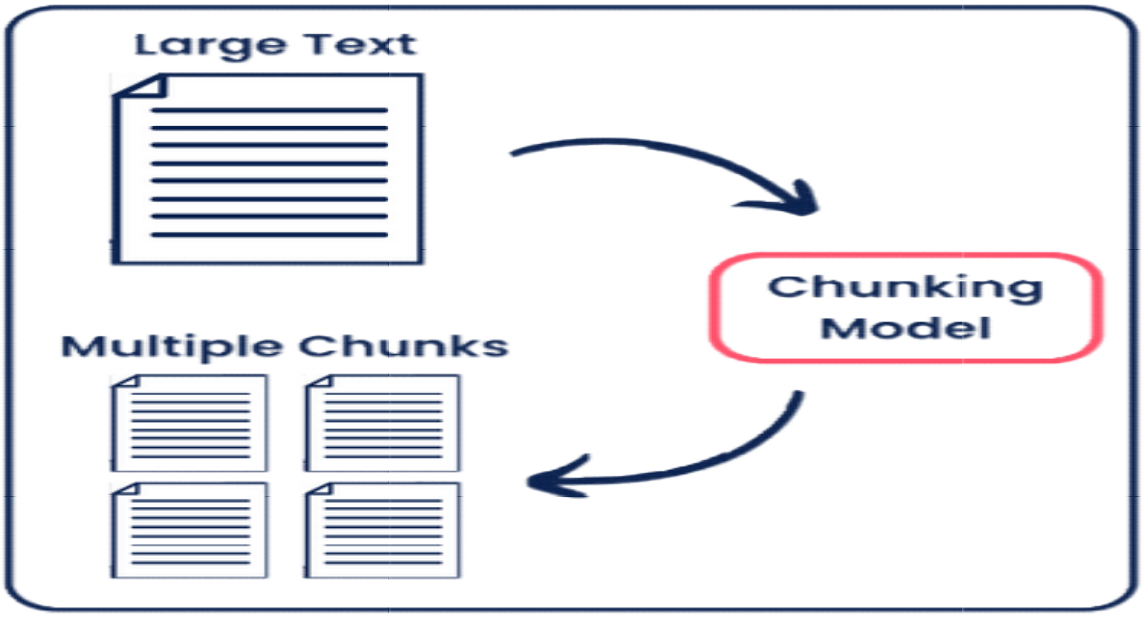
- **Purpose:**
  - ✓ To improve processing speed and accuracy by providing smaller, focused units of data for models to analyze.

- **Application in NLP:**
  - ✓ Particularly important in Natural Language Processing (NLP) tasks where models need to handle large text documents, like in retrieval-augmented generation (RAG) systems.

- **Benefits:**
  - ✓ **Efficient retrieval:** Smaller chunks allow for faster and more relevant information retrieval from large datasets.
  - ✓ **Context preservation:** By carefully splitting text, chunks can maintain the semantic meaning within each segment.
  - ✓ **Handling token limits:** Large language models often have token limitations, so chunking ensures that the input fits within processing constraints.

1. **How chunking works:**
   - ✓ **Splitting criteria:**

- ✓ Depending on the application, chunks can be created based on sentence boundaries, paragraph breaks, topic shifts, or a fixed number of words/tokens.
- ✓ **Overlapping chunks:**
- ✓ Sometimes, chunks can overlap slightly to ensure that important context isn't lost at the boundaries.



Types of Chunking

- There are, broadly, two types of chunking:

1. Chunking up
2. Chunking down

Chunking up:

- ✓ Here, we don’t dive deep; instead, we are happy with just an overview of the information. It just helps us get a brief idea of the given data.

Chunking down:

- ✓ Unlike the previous type of chunking, chunking down helps us get detailed information.
- ✓ So, if you just want an insight, consider “chunking up” otherwise prefer “chunking down”

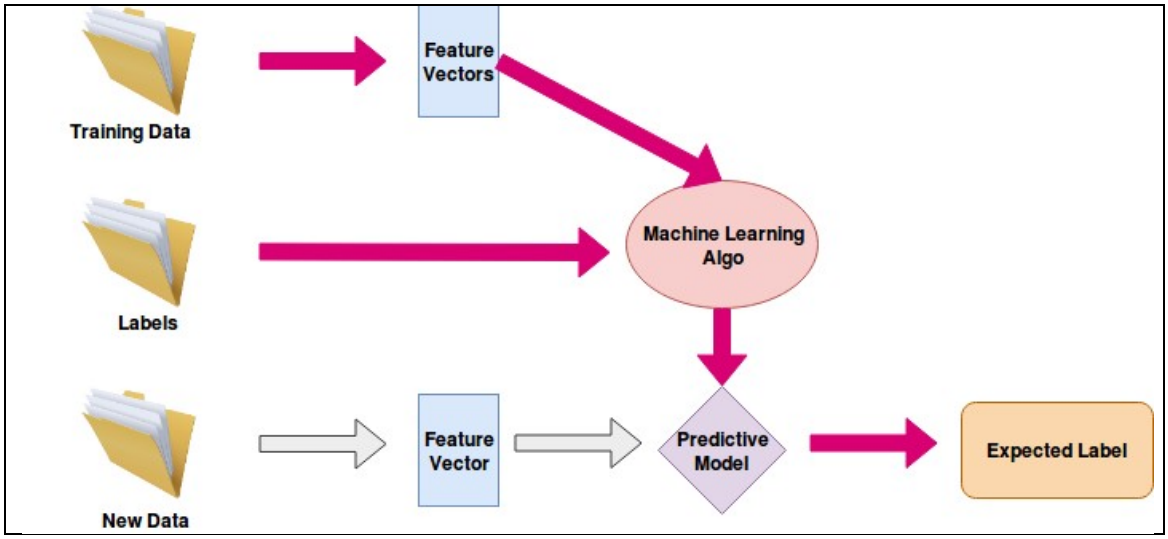
Text Classification

What is Text Classification?

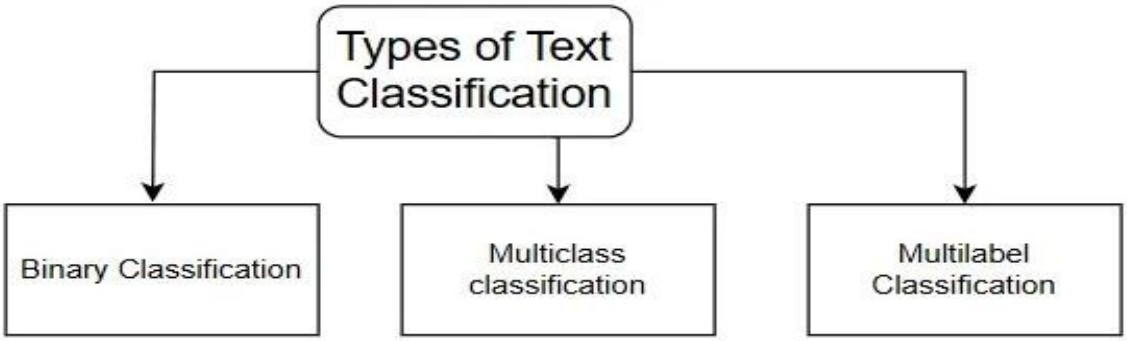
- ✓ Text classification is a fundamental task in natural language processing (NLP) that involves assigning predefined categories or labels to text documents.
- ✓ This process enables the automated sorting and organization of textual data, facilitating the extraction of valuable information and insights from large volumes of text. Text classification is widely used in various applications, including sentiment analysis, spam detection, topic labelling, and document categorization.

Why Use Scikit-learn for Text Classification?

- ✓ **Ease of Use**: User-friendly API and comprehensive documentation make it accessible for beginners and experts alike.
- ✓ **Performance**: Optimized for large datasets and efficient computation with robust model evaluation tools.
- ✓ **Integration**: Seamless integration with NumPy, SciPy, and pandas, plus support for creating streamlined workflows with pipelines.
- ✓ **Community Support**: Large, active community and frequent updates ensure continuous improvement and extensive resources for troubleshooting.



There are 3 types of text classification



1. **Binary classification :**
   Email spam classifier
2. **Multi class classification :**
   Reading a news article and then classifying into its class based on genre
3. **Multi label classification :**
   There can be multiple outputs for a single classification. Like a single news article can be classified into different classes. For example: A bollywood news article can be classified into classes like glamour, trending, youth, celebrity news etc