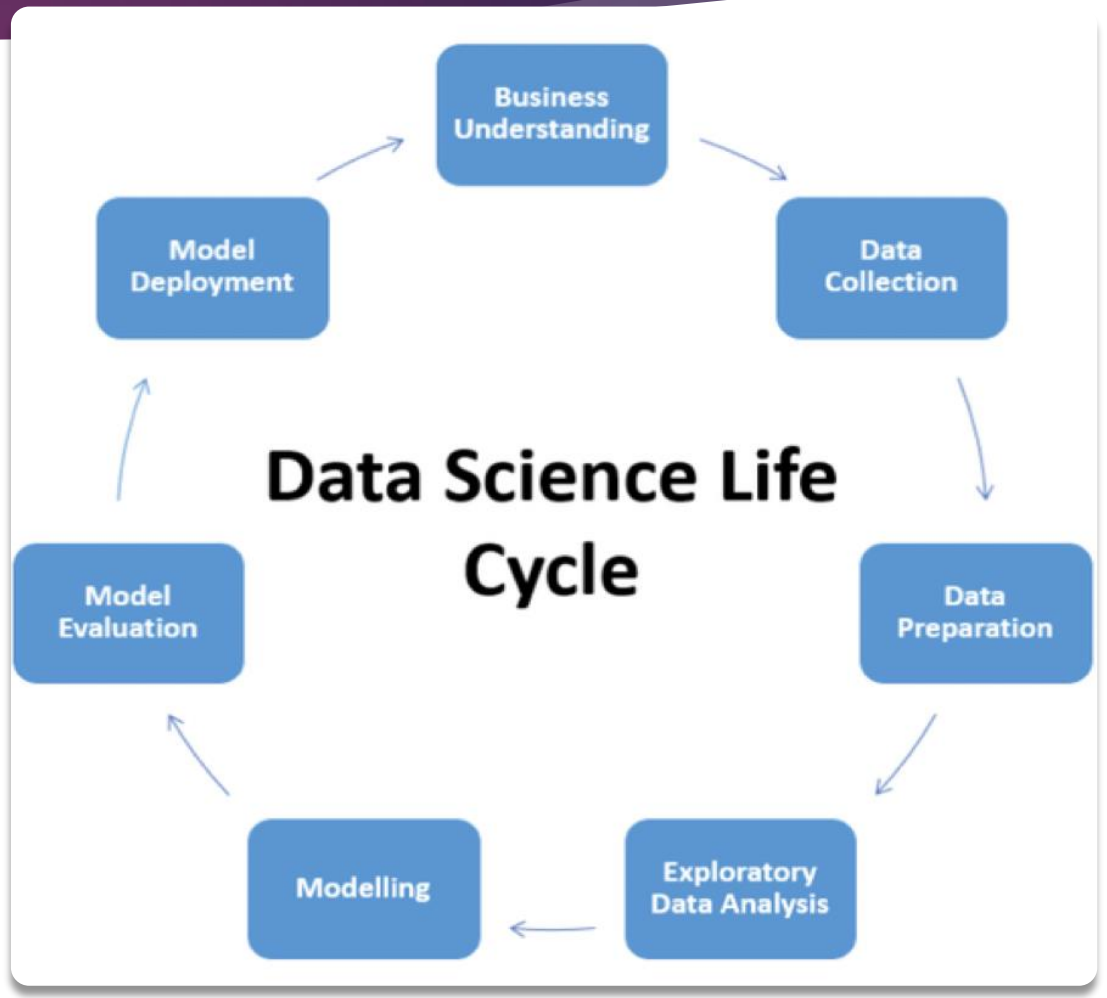
# Data Science

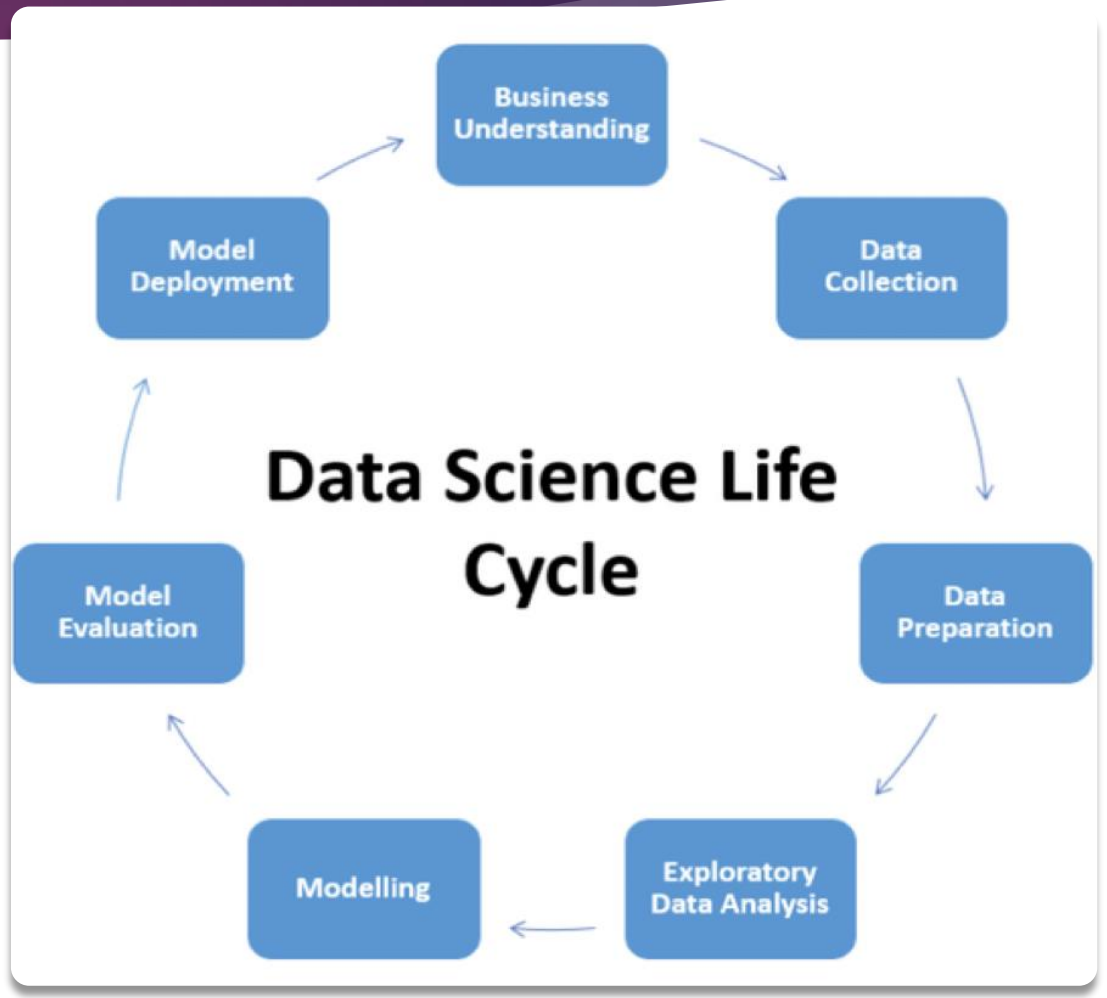
# Data Science Project life cycle

- Business Understanding
- Data Collection
- Data Preparation
- Exploratory data analytics(EDA)
- Data Modelling
- Model Evaluation
- Model Deployment



# Data Science Project life cycle

- Business Understanding
- Data Collection
- **Data Preparation**
- Exploratory data analytics(EDA)
- Data Modelling
- Model Evaluation
- Model Deployment



# Pre-processing text data

- Cleaning up the text data is necessary to highlight attributes that we are going to want our model to pick up on. Cleaning (or pre-processing) the data typically consists of  number of steps:
  1. **Removing  punctuation**
  2. **Converting text to lowercase**
  3. **Tokenization**
  4. **Removing stop-words**
  5. **Lemmatization /Stemming**
  6. **Vectorization**
  7. **Feature Engineering**

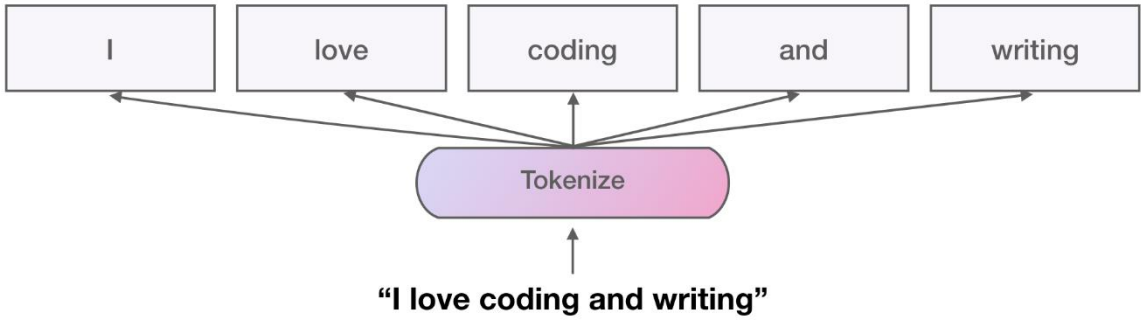
# Pre-processing text data

- Cleaning up the text data is necessary to highlight attributes that we are going to want our model to pick up on. Cleaning (or pre-processing) the data typically consists of  number of steps:
  1. **Remove punctuation**
  2. **Converting text to lowercase**
  3. **Tokenization**
  4. **Remove stop-words**
  5. **Lemmatization /Stemming**
  6. **Vectorization**
  7. **Feature Engineering**

# Pre-processing text data

- Cleaning up the text data is necessary to highlight attributes that we are going to want our model to pick up on. Cleaning (or pre-processing) the data typically consists of  number of steps:
  1. Remove punctuation
  2. Converting text to lowercase
  3. **Tokenization**
  4. Remove stop-words
  5. Lemmatization /Stemming
  6. Vectorization
  7. Feature Engineering

# Tokenization




- **Tokenization** is one of the most common tasks when it comes to working with text data
- Tokenization is essentially **splitting a phrase, sentence, paragraph, or an entire text document into smaller units**, such as individual words or terms. Each of these smaller units are called tokens.
- It is important because the meaning of the text could easily be interpreted by **analyzing the words present in the text.**
- Methods to Perform Tokenization in Python :
  - ✓ Tokenization using Python's **split()** function
  - ✓ Tokenization using Regular Expressions (**RegEx**)
  - ✓ Tokenization using **NLTK**
  - ✓ Tokenization using the other libraries like **spaCy and Gensim library**

- The final Goal of Tokenization is : **Creating Vocabulary**

# Pre-processing text data

- Cleaning up the text data is necessary to highlight attributes that we are going to want our model to pick up on. Cleaning (or pre-processing) the data typically consists of  number of steps:
  1. Remove punctuation
  2. Converting text to lowercase
  3. Tokenization
  4. **Remove stop-words**
  5. Lemmatization /Stemming
  6. Vectorization
  7. Feature Engineering

# Remove stop-words

- **Stopwords** are common words that are present in the text but generally do not contribute to the meaning of a sentence. They hold almost **no importance for the purposes of information retrieval** and natural language processing. They can safely be ignored without sacrificing the meaning of the sentence. For example – 'the' and 'a'.
- **Stop Words:** A stop word is a commonly used word (such as "**the", "a", "an", "in")** that a search engine has been programmed to ignore, both when indexing entries for searching and when retrieving them as the result of a search query.
- The NLTK package has a separate package of stop words that can be downloaded. NLTK has stop words **more than 16 languages** which can be downloaded and used. Once it is downloaded, it can be passed as an argument indicating it to ignore these words.
- `import nltk  from nltk.corpus`
- `import stopwords  set(stopwords.words('english'))`

# Remove stop-words

- If the concern is with **the context (e.g. sentiment analysis)** of the text it might make sense to treat words differently. For example, "**NOT**" is included as stop word but when considering context of text, negation changes the so-called *valence* of a text. This needs to be treated carefully and is usually not trivial.
- Considering example for **'not'**-
  - NLTK is a useful tool => NLTK useful tool
  - NLTK is not a useful tool => NLTK useful tool

# Pre-processing text data

- Cleaning up the text data is necessary to highlight attributes that we are going to want our model to pick up on. Cleaning (or pre-processing) the data typically consists of  number of steps:
  1. Remove punctuation
  2. Converting text to lowercase
  3. Tokenization
  4. Remove stop-words
  5. **Lemmatization /Stemming**
  6. Vectorization
  7. Feature Engineering

# Stemming

- **Stemming.** Is the process of reducing inflected or derived words to their **word stem or root**.
- Stemming is aiming to reduce variations of the **same root word.**

| | original_word | stemmed_words |
|---|---|---|
| 0 | connect | connect |
| 1 | connected | connect |
| 2 | connection | connect |
| 3 | connections | connect |
| 4 | connects | connect |

# Stemming in Example

- If Python sees **play**, **playing**, **played** and **plays** as four different separate things, that means:
  - it has to keep those Four separate **words in memory**. Imagine every variation of every root word. Maybe **we have a thousand root words.**
  - The alternative in this **play**, **playing**, **played** and **plays** example is applying the **stemmer**, all different words will become one word ; **play**,
  - Python has to look at a lot more tokens without a stemmer and it doesn't know that these separate tokens are even related.
  - In this case, we're not leaving it up to Python. We're being explicit by replacing similar words with just one common root word.
    - reduce the corpus of words that exposed to the model
    - Explicitly correlate the word with similar meaning

# Stemming approach

- Stemming is a **rule-based approach** because it slices the inflected words from prefix or suffix as per the need using a set of commonly underused prefix and suffix, like **“-ing”, “-ed”, “-es”, “-pre**”, etc. It results in a word that is actually not a word.
- The process of stemming means often **crudely chopping off the** end of a word, to **leave only the base**.
- So this means taking words with various **suffixes and condensing them under the same root word.**
  - There are mainly two errors that occur while performing Stemming, **Over-stemming**, and **Under-stemming.**
    - **Over-steaming** occurs when two words are stemmed from the same root of different stems.
      - Universal(عالمي)
      - university(جامعة)
      - universe(كون)
    - **Under-stemming** occurs when two words are stemmed from the same root of not a different stems
      - alumnus
      - alumni
      - alumnae

# Different types of stemmers

There are English and Non-English Stemmers available in **nltk** package.

- **Porter Stemmer**: PorterStemmer is known for its simplicity and speed. PorterStemmer uses **Suffix Stripping** to produce stems
- **Lancaster Stemmer**: LancasterStemmer is simple, but heavy stemming due to iterations and over-stemming may occur
- **The Snowball Stemmer**
- **Regex-Based Stemmer**.

# Different types of stemmers

There are English and Non-English Stemmers available in **nltk** package.

- **Porter Stemmer**: PorterStemmer is known for its simplicity and speed. PorterStemmer uses **Suffix Stripping** to produce stems
- **Lancaster Stemmer**: LancasterStemmer is simple, but heavy stemming due to iterations and over-stemming may occur
- **The Snowball Stemmer**
- **Regex-Based Stemmer**.

# lemmatizing

- Lemmatizing : **The process of grouping together the inflected forms of a word so they can be analyzed as a single term.**
- **Lemmatization,** unlike Stemming, reduces the inflected words properly ensuring that the root word belongs to the language
- lemmatizing is using **vocabulary analysis** of words to remove inflectional endings and return to the dictionary form of a word.
- So again : **play**, **playing**, **played** and **plays** would all be simplified down to **play**, because that's the root of the word. Each variation carries the same meaning just with slightly different tense.
- Will use the **WordNet lemmatizer**. This is probably the most popular lemmatizer.
- **WordNet** is a collection of nouns, verbs, adjective and adverbs that are grouped together in sets of synonyms, each expressing a distinct concept.
- This lemmatizer **runs off of this corpus of synonyms**, so given a word, it will track that word to its synonyms, and then the distinct concept that that group of words represents.

# lemmatizing Vs stemming

- The goal of both is to condense derived words down into their base form, **to reduce the corpus of words that the model's exposed to**, and to **explicitly correlate words with similar meaning**.
- There is an **accuracy and speed trade-off that** you're making when you opt for one over the other.
- The difference is that stemming takes a more crude approach by just **chopping off the ending of a word using heuristics,** without any understanding of the context in which a word is used. Because of that, stemming may or may not return an actual word in the dictionary. And it's usually **less accurate**,
- but the benefit is that it's **faster because** the rules are quite simple. Lemmatizing leverages more informed analysis to create groups of words with similar meaning based on the context around the word, part of speech, and other factors.
- Lemmatizers will always **return a dictionary word**. And because of the additional context it's considered, this is typically more accurate. But the downside is that it may be more computationally expensive.
- **The selection of Stemming or Lemmatization is solely dependent upon project requirements.** Lemmatization is mandatory for critical projects and projects where sentence structure matter like language applications

# Pre-processing text data

- Cleaning up the text data is necessary to highlight attributes that we are going to want our model to pick up on. Cleaning (or pre-processing) the data typically consists of  number of steps:
  1. Remove punctuation
  2. Converting text to lowercase
  3. Tokenization
  4. Remove stop-words
  5. Lemmatization /Stemming
  6. **Vectorization**
  7. Feature Engineering

# Vectorizing

- **Vectorizing** : The process that we use to convert text to a form that Python and a machine learning model can understand .
- It is defined as the process of **encoding text as integers to create feature vectors**.
- **A feature vector** is an **n-dimensional vector of numerical features that represent some object**. So in our context, that means we'll be taking an individual text message and converting it to a numeric vector that represents that text message.