Implementation(code):  
  
import pandas as pd

import re

import nltk

nltk.download('stopwords')

from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize

# Load the CSV into a DataFrame

df = pd.read\_csv('./Instructions2.csv')

# Text cleaning function

def clean\_text(text):

# Convert text to lowercase

text = text.lower()

# Remove special characters and numbers

text = re.sub(r'[^a-zA-Z\s]', '', text)

# Tokenize the text

tokens = word\_tokenize(text)

# Remove stopwords

stop\_words = set(stopwords.words('english'))

tokens = word\_tokenize(text)

tokens = [word for word in tokens if word not in stop\_words]

# Join the tokens back into a single string

cleaned\_text = ' '.join(tokens)

return cleaned\_text

# Apply the cleaning function to the 'Text' column

df['Cleaned\_Text'] = df['Text'].apply(clean\_text)

# Display the cleaned DataFrame

print(df.head())

As I tackled the text-cleaning process, I decided to incorporatecertaintechniques while leaving out others based on the nature of the data and my specific goals.

Here's my view for each step:

**INCLUDED:**

**1. Case Conversion:**

I ensured that all text was converted to lowercase because I wanted to maintain consistency in the text data.  
Considering words with various cases as identical streamlines following steps in processing and prevents duplicate words with different cases.

**4. Stop-words:**

I decided to remove this from the text data as these common words(e.g., "the", "is" and "and")often don't contribute much meaning to the text and can be eliminated without changing the meaning of the text data.

Removing this helps to focus on more meaningful words and improves the quality of the text data for analysis thus it makes it easy to read through and follow besides driving the meaning intended home.

**NOT INCLUDED:**

**5. Stemming:**

Stemming is the process of reducing words to their base form.

I chose not to implement this method of cleaning datasets due to concerns about its potential to eliminate required letters(such as 'e')from words, it dropped the vowels 'e' and 'a' from many words whenever they occurred at the end.

Since this could affect the readability and meaning of the text, I opted against implementing and using stemming in this cleaning process.

**6. Lemmatization**:

While this produces valid words by considering the context thus being better than stemming. It's computationally more intensive meaning it will require more compute time for a larger text data input.

This is not acceptable since it may lead to high computational costs that's why I chose not to include it in the cleaning process.

**REPORT:**

**Text Cleaning Methods Comparison Report**

This report covers the different methods of cleaning text as implemented and tested with the dataset with text data for analysis.

The objective is to test out various cleaning methods and explain why the technique selected was chosen, taking into account how efficient and appropriate the chosen method is for the job.

**Justification:**

I chose to include the following in the text-cleaning process:

* case conversion
* stopwords.

These methods proved effective in improving the quality and readability of the text and prepared the text for more analysis.

Stemming and lemmatization were not implemented as they altered the meaning of the text, lemmatization was computationally intensive and thus could not be a better option as optimal performance is key. Correction of misspelled words was not implemented as there were no significant instances of misspellings in the dataset.

In a nutshell, the chosen cleaning processes were selected based on:

* Effectiveness
* Suitability
* Computational efficiency.

By applying:

* Case conversion
* Stopwords removal

The dataset was successfully cleaned of anything that may alter the meaning of the text data or make it hard to follow, this ensures consistency and improved quality of the text data for any further analysis.