Assignment 1.

General.
The CSV file and txt file need to be uploaded in their respective cells.

Task1.
Assumption. None
Method.

1. Split the message by the spaces.
2. Check the first letter of each word with a list of letters containing vowels and consonants.

Execution.

1. Execute the tab for running the method on the entire dataset.
2. To run on a single sentence, uncomment the line of code marked and add the sentence.

Task 2.
Assumption.
Capitalized words are words with all letters capitalized. All characters are letters from the English alphabet.

Method.

1. Split the message by the spaces.
2. Check each letter of each word, if all the letters are capital then consider capitalized words.

Execution.

3. Execute the tab for running the method on the entire dataset. (Uncomment the line of code to print the capitalized words.)
4. To run on a single sentence, uncomment the line of code marked and add the sentence.

Task 3.
Assumption.
Email ids are of the form x.y...@z.com
Phone numbers are 10 digits long, may use calling codes of a maximum of 2 digits, and may have (+) at the beginning of the number. May have spaces or hashes or dots in the usual format.

Method.

1. Defined a regex for emails and phone numbers.
2. Loop through all the messages and use re.findall() to get all occurrences of the phone numbers and email ids.
3. Printed the following values.
   The number of email ids (phone numbers) is.
   The percentage of phone numbers or email Ids present in spam(ham) messages:
   Formula = (Emails And Numbers in Spam/(Total Email Ids + Total Phone numbers))*100
   The percentage of spam(ham) messages with email Ids or phone numbers
   (Number of Spams with email ids or phone numbers /Total number of Spams)*100

Execution.

1. Execute the tab for running the method on the entire dataset.
2. To run on a single sentence, uncomment the line of code marked and add the sentence.

Task 4.
Assumption.
May start with the following characters. Rs.,$,£,€,¥,₹ or may end with INR, €, pp, p_. Must include some length of number, that is not constrained.
It includes pp for “ppm that is paise per minute or pps paise per second)and p_ for “p per minute, p daily and so on”

Method.

1. Defined a regex for monetary quantities.
2. Loop through all the messages and use re.findall() to get all occurrences of the monetary quantities.
3. Printed the following values.
   The number of monetary quantities is.
   The percentage of monetary quantities present in spam(ham) messages:
   Formula = (monetary quantities in Spam/(Total monetary quantities))*100
   The percentage of spam(ham) messages with monetary quantities
   (Number of Spams with monetary quantities /Total number of Spams)*100

Execution.

1. Execute the tab for running the method on the entire dataset.
2. To run on a single sentence, uncomment the line of code marked and add the sentence.

Task 5.
Assumption.

None

Method.

1. Defined a regex for emoticons. (Same as defined by tweet tokenizer)
2. Tokenize the message using the regex tokenizer with the given regex.

Execution.

1. Execute the tab for running the method on the entire dataset.
2. To run on a single sentence, uncomment the line of code marked and add the sentence.

Task 6.
Assumption.
None

Method.

1. Defined a regex for words with clitics.
2. Use the regex on the message with re. match()

Execution.

1. Execute the tab for running the method on the entire dataset.
2. To run on a single sentence, uncomment the line of code marked and add the sentence.

Task 7.
Assumptions.
The word in the message can be attached to punctuation marks, that is, punctuation marks surrounding the words will be ignored.
The function is case-insensitive
The given word does not include any punctuation marks.

Method.

1. Tokenize the message using regex tokenize.
2. Compare the first word with the given word.

Execution.

1. Execute the tab for running the method on the entire dataset. Enter the word in the space.
2. To run on a single sentence, uncomment the line of code marked and add the sentence.

Task 8.
Assumptions.
The word in the message can be attached to punctuation marks, that is, punctuation marks surrounding the words will be ignored.
The function is case-insensitive
The given word does not include any punctuation marks.

Method.

1. Tokenize the message into sentences.
2. Tokenize the sentences into words.
3. Compare the last word with the given word.

Execution.

1. Execute the tab for running the method on the entire dataset. Enter the word in the space.
2. To run on a single sentence, uncomment the line of code marked and add the sentence.

Task 9.
Assumption. None
Method.

1. Calculate the required values for the heuristics.
2. Use the manually implemented heuristic to assign category values based on the above functions.
3. Calculate the accuracy using the actual labels.

Execution

1. Run the cell to print the 7 accuracies and the winning heuristic number.

Task 10.
Assumption.
Upload the text file with name text.txt. (Or change at its place).

The word in the message can be attached to punctuation marks, that is, punctuation marks surrounding the words will be ignored.
The function is case-insensitive
The given word does not include any punctuation marks.

Method.

1. Load the text document.
2. Tokenize the data into sentences.
3. Tokenize the sentences into words.
4. Compare each word to the given word.

Execution.

1. Execute the tab for running on the “text.txt” file. Enter the custom word.
2. Uncomment the comment to print the sentences with the required word.

.