This is an explanation of the Python code of SMS text data feature extracting using the Bag of Word (BoW) model. In this code, I have defined nine functions. The nine functions are:

(i) load_data()
(ii) preprocess_text()
(iii) remove_punctuations_digits()
(iv) tokenize_text()
(v) remove_stopwords()
(vi) train_bow_model()
(vii) print_feature_names()
(viii) convert_bow_matrix_to_df()
(ix) main()

In load_data() function I have read a CSV file named 'SMSSpamCollection' using pandas (pd). It sets the separator as a tab ('\t'), assumes no header in the file (header=None), and assigns column names as "Class" and "SMS_Text".

In preprocess_text() function, I have created a new column 'PREPROCESSED_SMS' in the data_SMS DataFrame by (data_SMS[PREPROCESSED_SMS']), then convert the text in the 'SMS_Text' column of data_SMS to lowercase. For lowercase, I used the str.lower() method.

In remove_punctuations_digits() function, I removed punctuations and digits from the 'PREPROCESSED_SMS' column of data_SMS by the regular expression r'[^a-zA-Z\s]'. This is a regular expression pattern that matches any character that is not an uppercase or lowercase letter or whitespace with an empty string. '\s' stands for whitespace character. 'lambda x' is an anonymous function that takes an input x and uses the re.sub() function from the python re module to substitute. The pattern will extract punctuations snd digits from the text data.

In tokenize_text() function, I have tokenized the text in the 'PREPROCESSED_SMS' column of data_SMS using apply word_tokenize from NLTK.

In remove_stopwords() function, I have removed the stopwords from the 'PREPROCESSED_SMS' column of data_SMS using NLTK's English stopwords and a list comprehension. It iterates through each word in the input x and includes it in the new list if it is

not found in the StopwordsEnglish list. The stopwords removed list is saved in a new column ‘StopwordsRemoved’ of data_SMS.

In train_bow_model() function, I have created a Bag-of-Words model using CountVectorizer. It fits and transforms the 'StopwordsRemoved' column of data_SMS into a sparse matrix and returns both the matrix and the fitted vectorizer. The data_SMS['StopwordsRemoved'] contains lists of words after the removal of stopwords in each SMS text. The lambda function is applied to each element x in these lists of words. lambda x: ' '.join(x) takes each list of words x and uses join() to concatenate the words with a space in between, basically converting the list of words into a single string.

The print_feature_names() function prints the feature names generated by the given SMS_vectorizer.

In convert_bow_matrix_to_df() function, I have converted the Bag of Words matrix SMS_bow into a Pandas DataFrame using the feature names obtained from the SMS_vectorizer. It prints the DataFrame with column names: words and returns it.

In the main() function, the eight defined functions call load_data() which loads the dataset, preprocess_text(data_SMS) which perform case normalization (lower case) to the text data and stored in a new variable ‘preprocessed_data’, remove_punctuations_digits(preprocessed_data) which remove the punctuations and digits from the ‘preprocessed_data’ and stored in a new variable ‘punctuations_digits_removed_data’, tokenize_text(punctuations_digits_removed_data) which tokenize the ‘punctuations_digits_removed_data’ and stored in a new variable tokenized_data, remove_stopwords(tokenized_data) which remove the stopwords from ‘tokenized_data’ and stored in a new variable stopword_removed_data, train_bow_model(stopword_removed_data) which train the BoW Model on ‘stopword_removed_data’ and stored in two variables ‘SMS_bow’ and ‘SMS_vectorizer’, print_feature_names(SMS_vectorizer) which print out the names of the features of SMS Vectorized and convert_bow_matrix_to_df(SMS_bow, SMS_vectorizer) which convert my BoW matrix into Pandas DataFrame (column names: words) and stored in ‘bow_df’ variable in order to execute when the python code is run as the main program.