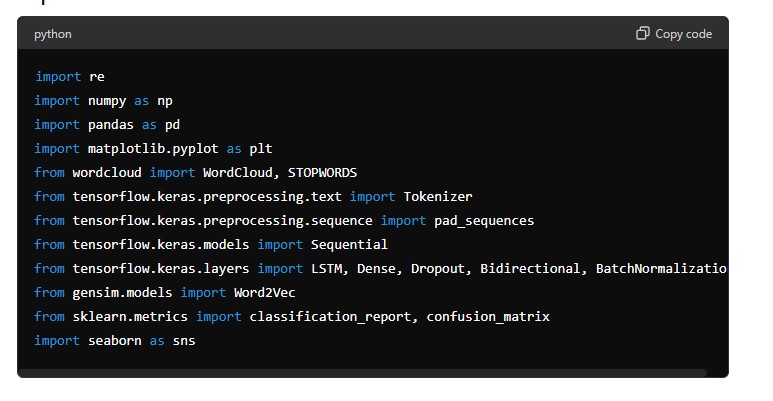
**SPAM OR HAM SMS CLASSIFICATION MODEL**

This document explains the implementation of a machine learning model for SMS spam classification using LSTM and Word2Vec. The model is built using Python and various libraries like TensorFlow, Keras, Gensim, and others. The following sections explain each part of the code in detail.



This block imports all necessary libraries and modules:

* **re** for regular expressions used in text preprocessing.
* **numpy** and **pandas** for numerical operations and data handling.
* **matplotlib.pyplot** and **seaborn** for visualization.
* **wordcloud** for generating word clouds.
* **tensorflow.keras** modules for building and training the LSTM model.
* **gensim.models.Word2Vec** for loading the pre-trained Word2Vec model.
* **sklearn.metrics** for evaluating the model.



These constants are used throughout the script:

* **drop\_value**: Dropout rate for regularization.
* **max\_len**: Maximum length of sequences.
* **trunc\_type** and **padding\_type**: Specify how sequences should be truncated and padded.
* **oov\_tok**: Token for out-of-vocabulary words.
* **num\_epochs**: Number of epochs for training the model.
* 

This function loads the data from a text file. Each line is split by tabs into labels and messages, and the data is returned as a Pandas DataFrame.

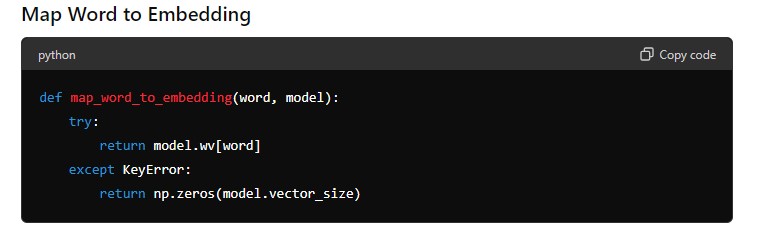
This function preprocesses a single message:

* Converts the message to lowercase.
* Removes URLs.
* Removes non-alphanumeric characters (except spaces, **$**, and **£**).

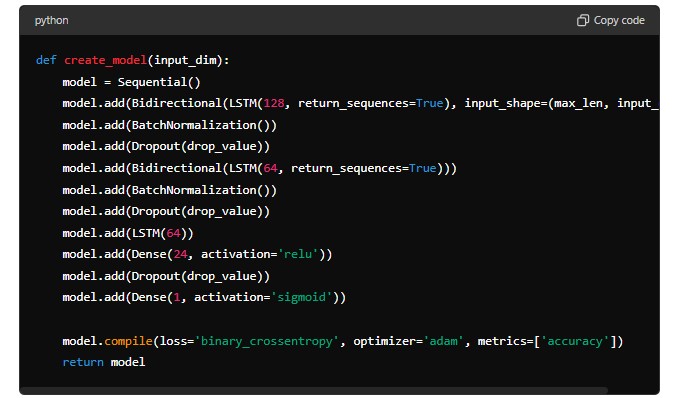


This function generates and displays word clouds for ham (non-spam) and spam messages:

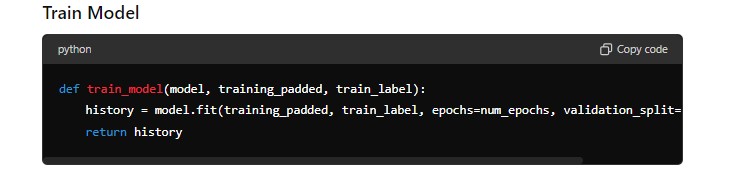
* Separates the messages into ham and spam.
* Joins messages into a single string for each category.
* Generates and displays word clouds using the **WordCloud** class.

This function maps a word to its embedding vector using a pre-trained Word2Vec model. If the word is not in the vocabulary, it returns a vector of zeros. This function preprocesses the data for training:

* Applies message preprocessing.
* Balances the dataset by sampling an equal number of ham and spam messages.
* Converts messages to sequences of word embeddings.
* Pads the sequences to ensure consistent length.

This function creates the LSTM model using Keras:

* Uses a **Sequential** model with several **Bidirectional LSTM**, **BatchNormalization**, and **Dropout** layers.
* The final layers are a dense layer and a sigmoid activation layer for binary classification.
* Compiles the model with binary cross-entropy loss and the Adam optimizer.

This function trains the model:

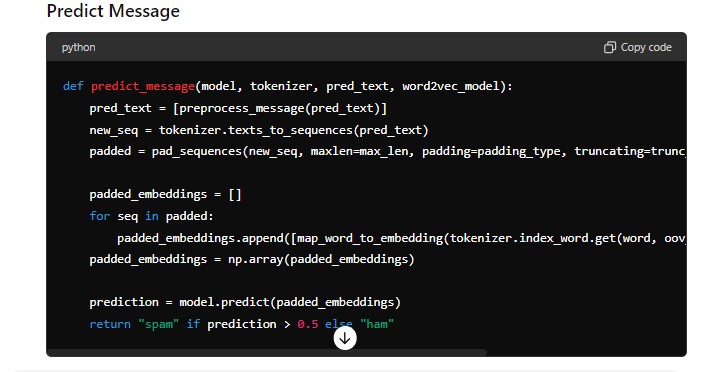
* Uses the **fit** method to train the model on the padded sequences and labels.
* Splits the training data into training and validation sets.
* Returns the training history.

This function evaluates the trained model:

* Predicts labels for the training data.
* Prints a classification report.
* Displays a confusion matrix using a heatmap.

This function predicts whether a given message is spam or ham:

* Preprocesses the message.
* Converts it to a sequence and pads it.
* Maps the sequence to word embeddings.
* Predicts the label using the trained model.
* Returns "spam" if the prediction is greater than 0.5, otherwise "ham".



This function predicts whether a given message is spam or ham:

* Preprocesses the message.
* Converts it to a sequence and pads it.
* Maps the sequence to word embeddings.
* Predicts the label using the trained model.
* Returns "spam" if the prediction is greater than 0.5, otherwise "ham".