Spam Mail Detection using Tensor Flow

Summary:

Email Spam has become a major problem nowadays, with Rapid growth of internet users, Email spams is also increasing. People are using them for illegal and unethical conducts, phishing and fraud. Sending malicious link through spam emails which can harm our system and can also seek in into your system. Creating a fake profile and email account is much easy for the spammers, they pretend like a genuine person in their spam emails, these spammers target those peoples who are not aware about these frauds. So, it is needed to Identify those spam mails which are fraud, this project will identify those spam by using TensorFlow and Keras This internship code confidently creates a binary text classification model using TensorFlow and Keras to differentiate between spam and non-spam (ham) messages. The primary objective of this project is to establish a solid foundation for a more robust spam detection system.

Dataset for Implementation:

This database contains 14 attributes listed below:

1. Unnamed
2. label
3. text
4. label_num

The numerical features consist of Unnamed: 0 ,which indicates the index or identifier for each row in the dataset. It doesn't seem to have a meaningful name and might have been automatically generated during data processing or extraction. It's often present in datasets but doesn't typically convey useful information about the data itself ,Label indicates whether the text message (or email) is classified as "ham" (not spam) or "spam" (unwanted or unsolicited messages, typically advertising). The label is assigned based on the content of the text ,text contains the actual text of the messages. It appears to be the main body of the messages, likely the content that is being classified as either "ham" or "spam",label_num seems to be a numerical representation of the "label" column. For example, it might have the value 0 for "ham" and 1 for "spam". This numerical

representation is often useful for machine learning algorithms that require numerical inputs.

Importing Libraries like:

1. Pandas *(for data manipulation)*
2. Matplotlib *(for data visualization)*
3. SkLearn *(for data modeling)*
4. Tensorflow *(for building and training neural networks)*

Data Preparation:

Cleaning and Handling Missing Values:

The initial step involved a thorough examination of the dataset to identify missing, inconsistent, or outlier data points. To find whether they are missing values in dataset or not we use the isnull() function and they are no missing values present in the dataset. Before that we use the info() and describe() function to find the attributes and Statistical values of the attributes.

Removing Unwanted Columns in dataset:

After cleaning and handling the missing values we remove the unwanted columns from the dataset. we drops the 'Unnamed: 0' and 'label' columns from the data frame. This will help clean up your dataset and remove unnecessary information.

Splitting the Data:

We partitioned the dataset into a 80:20 ratio, allocating 80% for training and 20% for testing. This split ensures sufficient data for learning while retaining a substantial subset for an unbiased evaluation of the model's performance.

Training the model by Tensorflow:

In this first ,we Import necessary libraries Tokenization and Padding. After that define the maximum length of sequences and create a Tokenizer object with an out-of-vocabulary token ("<OOV>"). We fit the Tokenizer on the training data (X_train).This step builds the vocabulary based on the text data. Then We converts it into sequences of integers, and pads the sequences to a maximum length for training a model. we are going to create a sequential model using TensorFlow and Keras. This model consists of an embedding layer, a flatten layer, and two dense layers. The embedding layer helps to represent the input data, the flatten layer flattens the input, and the dense layers perform computations on the flattened input. The last dense layer uses the sigmoid activation function, which is commonly used for binary classification tasks. This model is compiled using the Adam optimizer, binary cross-entropy loss, and accuracy as the evaluation metric. After that model is trained using the fit function with 5 epochs and a validation split of 0.2. The fit function trains the model on the provided training data (X_train_pad and Y_train) for the specified number of epochs. Predictions are made on the test data (x_test_pad) using model.predict(). Predictions are thresholded at 0.5 to obtain binary predictions. Finally, the binary predictions are displayed.

Conclusion:

This code confidently implements a basic text classification model using TensorFlow and Keras. With further refinement and evaluation, this code has the potential to serve as a strong foundation for building more robust spam detection systems.