A simple explanation of the code of 1st part:

Step 1: Upload the dataset

- The code uses `files.upload()` from Google Colab to upload a CSV file containing the dataset.

Step 2: Import necessary libraries

- The code imports libraries such as `pandas`, `scikit-learn`, and `joblib` to handle data manipulation, machine learning, and model saving.

Step 3: Load and clean the dataset

- The CSV file (`Dataset.csv`) is loaded into a pandas DataFrame (`data`).

- Any rows with missing data are removed with `dropna()`.

- The target variable (labels) in the dataset is encoded into numerical values using `LabelEncoder()` (i.e., converting words like "positive" or "negative" into numbers like 0 or 1).

Step 4: Split the data

- The dataset is split into training and testing sets using `train_test_split`. 80% of the data is used for training, and 20% is used for testing.

Step 5: Create a machine learning pipeline

- A pipeline is created that first converts text data (sentences) into numerical vectors using `TfidfVectorizer()`, then trains a Support Vector Machine (SVM) classifier (`LinearSVC()`) to classify the data.

- The model is trained on the training data (`X_train` and `y_train`).

Step 6: Predict and evaluate the model

- The trained model is used to predict the labels of the test data (`X_test`).

- The accuracy of the model is calculated and displayed.

- A classification report is generated, showing detailed metrics such as precision, recall, and F1 score for each class.

Step 7: Save the model and encoder

- The trained model and the label encoder are saved as `.pkl` files using `joblib.dump()`, so they can be loaded and used later for predictions.

In summary:

This code trains a machine learning model to classify sentences into predefined categories (labels), evaluates its performance, and saves the trained model and encoder for future use.

A simple explanation of each step in the code of 2nd part:

Step 1: Upload the PDF, model, and label encoder files

- This step uses `files.upload()` from Google Colab to upload the necessary files: a PDF document, the saved machine learning model (`sentence_classifier_pipeline.pkl`), and the label encoder (`label_encoder.pkl`).

Step 2: Install PyMuPDF (fitz) to handle PDF extraction

- The `!pip install pymupdf` command installs the `PyMuPDF` library (also known as `fitz`), which is used to extract text from PDF files.

Step 3: Import necessary libraries

- The code imports:

- `fitz` (PyMuPDF) for handling PDF file reading and text extraction.

- `joblib` to load the saved machine learning model and label encoder.

- `re` for using regular expressions to split the text into sentences.

Step 4: Load the trained model and label encoder

- The trained model (`sentence_classifier_pipeline.pkl`) and the label encoder (`label_encoder.pkl`) are loaded using `joblib.load()`.

- The model is used to classify sentences into categories.

- The label encoder helps to convert numerical labels back into the original category labels (e.g., converting numbers back into terms like "positive" or "negative").

Step 5: Extract text from the uploaded PDF

- The PDF file (named `SRS.pdf` here) is opened with `fitz.open()`, and all the text content is extracted from each page of the PDF using `page.get_text()`. The extracted text is stored in the variable `text`.

Step 6: Preprocess the extracted text into individual sentences

- The extracted text is split into individual sentences using the `preprocess_text()` function.

- This function uses regular expressions to detect sentence boundaries (periods or question marks followed by whitespace).

- It ensures that each sentence is cleaned up by removing any unnecessary whitespace.

Step 7: Classify each sentence and print the predicted labels

- For each sentence extracted from the PDF:

1. The `pipeline.predict()` method is used to predict the label of the sentence. The result is a numerical label.

2. The label is then converted back to its original category name using `label_encoder.inverse_transform()`.

3. The sentence and its predicted label are printed to the console.

In summary:

This code takes a PDF file, extracts the text from it, splits the text into individual sentences, classifies each sentence using a pre-trained machine learning model, and then prints the predicted labels for each sentence. The model can be used to categorize sentences in the PDF, such as identifying specific topics or sections from a Software Requirement Specification (SRS) document.