**To Build the model using layoutLM:**

1. **PDF Conversion and Data Preparation:**
   * The code begins with functions to convert PDF files to images and to list images along with their categories (labels).
   * The **convert\_pdf\_to\_images** function converts each page of a PDF file into an image (JPEG format) and saves it to the specified output directory.
   * The **list\_images\_and\_categories** function traverses through a root directory, finds PDF files, converts them into images, and collects image paths along with their corresponding categories (labels).
2. **OCR (Optical Character Recognition):**
   * After obtaining the images, OCR is applied to extract text and bounding boxes of words within each image. This is done using the Tesseract OCR engine.
   * OCR results (words and their corresponding bounding boxes) are normalized and added as extra columns to the dataset.
3. **Encoding for LayoutLM:**
   * Each example (image with associated text and bounding boxes) is encoded for LayoutLM model input.
   * Encoding involves tokenizing the text, truncating the tokens if necessary, and padding to a maximum sequence length.
   * Bounding box coordinates are padded accordingly and added as an extra feature to the encoding.
4. **Dataset Preparation and DataLoader:**
   * Features for the dataset are defined, including input\_ids, attention\_mask, token\_type\_ids, bbox, label, image\_path, and words.
   * The encoded dataset is created, and the format is set to 'torch' for PyTorch compatibility.
   * PyTorch DataLoader is used to create batches of data for training.
5. **Model Definition and Training:**
   * LayoutLM For Sequence Classification model is initialized for sequence classification with a specified number of labels.
   * AdamW optimizer is used for training.
   * The model is trained in a loop over multiple epochs. In each epoch, the data is iterated over, and for each batch, the model is trained, and gradients are updated.
   * Loss and training accuracy are printed at the end of each epoch

To predict the new Document:

This script seems to aim at predicting the label (category) of a given PDF file using a pre-trained LayoutLM model. Here's an overview of how it works:

1. **Model Loading:**
   * The function **load\_model** loads a pre-trained LayoutLM model and tokenizer from the specified directory. It also determines the device to use (CPU or GPU).
2. **PDF Preprocessing:**
   * The function **preprocess\_new\_pdf\_images** converts each page of the PDF file into an image.
   * OCR (Optical Character Recognition) is applied to extract text from each image.
   * Bounding boxes for each word are extracted using OCR data.
3. **Encoding for Model Input:**
   * The function **encode\_example** encodes each example (image with associated text and bounding boxes) for model input.
   * Tokenization is performed on the text, and bounding boxes are normalized and padded.
   * The encoded examples are converted into PyTorch tensors.
4. **Inference:**
   * The function **predict\_label** takes the preprocessed data, encodes it, and runs inference using the loaded model.
   * It returns the predicted label for the input PDF file.
5. **Usage:**
   * The model is loaded using **load\_model**.
   * Input PDF file path is provided to **predict\_label** function to get the predicted label.