Fraud Email using BERT Classification

Detailed Breakdown of Functions and Their Documentation

1. FocalLoss Class

Purpose:
The `FocalLoss` class is a custom loss function designed to address class imbalance by focusing more on difficult or misclassified examples during training.

Methods:

- `__init__(self, alpha=0.25, gamma=2.0, reduction='mean')`
  Description:
  Constructor to initialize the focal loss with configurable parameters.
  Parameters:
  - `alpha` (float): Weighting factor for the loss of each class (default 0.25).
  - `gamma` (float): Focusing parameter to reduce the loss contribution from easy examples (default 2.0).
  - `reduction` (str): Specifies the reduction to apply to the output (`'mean'`, `'sum'`, or `'none'`).
- `forward(self, inputs, targets)`
  Description:
  Computes the focal loss given the model's raw predictions (`inputs`) and the true labels (`targets`).
  Parameters:
  - `inputs` (Tensor): Model predictions (logits) for each class.
  - `targets` (Tensor): Ground truth class labels.
    Returns:
  - A Tensor representing the computed focal loss based on the reduction specified during initialization.

2. Additional Functions and Classes

(Note: The notebook further defines functions for data processing, model training, and evaluation. The following is a summary of these components as derived from the notebook.)

a. Data Processing Functions

These functions include:

- **Dataset Loader Class (Custom Dataset)**
  Purpose:
  A custom `Dataset` class (e.g., `EmailDataset`) is implemented to handle tokenization and conversion of email texts into the format required by the BERT model.
  Documentation:
  - **`__init__(self, dataframe, tokenizer, max_length)`**:
    Initializes the dataset with a pandas DataFrame containing the email data, a tokenizer instance, and a maximum sequence length for BERT.
  - **`__len__(self)`**:
    Returns the total number of samples.
  - **`__getitem__(self, index)`**:
    Retrieves the tokenized email and its corresponding label for the given index.

b. Model Training and Evaluation Functions

Functions here facilitate training and evaluating the model:

- **Training Function (e.g., `train_model`)**
  Purpose:
  Runs the training loop for the model, iterating over batches of training data, computing loss (using focal loss or cross-entropy), and updating model weights via backpropagation.
  Documentation:
  - **`train_model(model, dataloader, optimizer, scheduler, loss_fn, device)`**:
    Trains the BERT model for one epoch.
    Parameters:
    - `model`: The BERT model instance.

- `dataloader`: A DataLoader object containing the training data.
- `optimizer`: Optimizer for updating model parameters (e.g., AdamW).
- `scheduler`: Learning rate scheduler (e.g., cosine schedule with warmup).
- `loss_fn`: Loss function (custom `FocalLoss` or another).
- `device`: Device (CPU/GPU) on which to run the training. **Returns:**
- The average training loss for the epoch.

- **Evaluation Function (e.g., `evaluate_model`)**

  Purpose:

  Evaluates the performance of the model on a validation or test set.

  Documentation:

  - **`evaluate_model(model, dataloader, loss_fn, device)`**:

    Computes validation loss and other metrics (accuracy, F1-score, precision, recall).

    Parameters:

    - `model`: The BERT model instance.
    - `dataloader`: A DataLoader object containing the evaluation data.
    - `loss_fn`: Loss function used for evaluation.
    - `device`: Device (CPU/GPU) for evaluation. **Returns:**
    - A dictionary containing evaluation metrics and loss.

c. Data Augmentation Function

- **Augmentation using EDA (Easy Data Augmentation)**

  Purpose:

  Applies augmentation techniques (synonym replacement, random insertion, etc.) to enrich the dataset and improve model robustness.

  Documentation:

  - **`augment_text(text)`**:

    Augments a given text string using EDA.

    Parameters:

    - `text` (str): The input email text. **Returns:**
    - A new text string that has been augmented.