Project Code Description

This document provides an overview of the code structure and descriptions for the main functionalities implemented in the final project. The project focuses on classifying mathematical problems by type using machine learning techniques, specifically logistic regression and BERT models.

1. logistic_regression.py

Description: This script implements a Logistic Regression model to classify mathematical problems based on textual descriptions. It uses the TfidfVectorizer for feature extraction and logistic regression for classification.

Functionalities:

- load_data(file_path): Loads data from a JSON file. Prints a success message if the data is loaded correctly or an error message if the loading fails.
- prepare_features(vectorizer, data, text_column='problem'): Transforms problem descriptions into numerical features using TF-IDF vectorization.
- main(): Orchestrates the loading of training and testing data, feature preparation, model training, prediction, and evaluation. Outputs the model's accuracy and classification report.

2. combine_problems.py

Description: This script is responsible for combining individual JSON files containing mathematical problems into two large JSON files, one for training and one for testing. This preprocessing step facilitates easier manipulation and access during model training and evaluation.

Functionalities:

- combine_json_files(root_dir, output_file): Traverses directories containing JSON files, combines them into a single list, and writes this list to an output file. Designed to handle both training and testing datasets.

3. bert.py

Description: This script utilizes the BERT model (bert-base-uncased) from Hugging Face's Transformers library to classify mathematical problems. It is an advanced approach using a pre-trained transformer model which is fine-tuned for this specific classification task.

Functionalities:

- load_data(file_path): Loads data from a JSON file into a Pandas DataFrame and converts it to a Hugging Face Dataset.
- tokenize_data(example): Tokenizes problem descriptions, preparing them for input into the BERT model.

- save_model(model, tokenizer, model_path): Saves the trained model and tokenizer to specified paths.
- transform_labels(example): Transforms categorical labels into numerical labels.
- Trainer Setup and Execution: Configures and executes training and evaluation of the BERT model, including setting training arguments, initializing the trainer, and performing training and evaluation.

4. predict.py

Description: This script is designed to use a pretrained BERT model to predict the categories of mathematical problems from textual descriptions. It leverages the Hugging Face Transformers library for model deployment and predictions.

Functionalities:

- load_model(model_path): Loads a pretrained BERT model and tokenizer from the specified path, setting the model to evaluation mode.
- predict(model, tokenizer, texts, batch_size=10): Processes batches of text, using the BERT model to predict the category of each text. It returns a list of predicted category indices.
- load_data(file_path): Loads mathematical problems from a JSON file, extracting problems and their corresponding labels.
- Main Execution Block: Loads the model, tokenizer, and test data, then makes predictions on the test data. The script subsequently saves the predicted labels alongside the original data into a new JSON file, predictions.json.

5. results.py

Description: This script evaluates the performance of the model predictions by generating a classification report that includes precision, recall, and F1-scores for each category.

Functionalities:

- load_predictions(file_path): Loads the predicted and actual labels from a JSON file (predictions.json).
- Main Execution Block: Calls load_predictions to retrieve labels, computes the classification report using scikit-learn's classification_report function, and prints the report to the console. This provides a detailed breakdown of the model's performance across different categories.