**LSTM Engine Stability Classification Model Documentation**

**Overview:**

This LSTM model is designed to classify engine stability based on time series data collected from snapshots of engine features. The model is trained on two categories of engine data: stable and unstable. It learns to distinguish between stable and unstable engine behavior by analyzing sequential data of engine features.

**Model Architecture:**

The LSTM (Long Short-Term Memory) model architecture consists of one LSTM layer followed by a Dense layer with a sigmoid activation function. Here's a detailed breakdown of the architecture:

1. **LSTM Layer:**
   * Units: 50
   * Input shape: (401, 11)
   * Activation function: Linear
2. **Dense Layer:**
   * Units: 1
   * Activation function: Sigmoid

A diagram of a computer

Description automatically generated

**Data Preprocessing:**

1. **Loading Data:**
   * The model loads engine data from Excel files stored in the "stable" and "unstable" folders.
   * Each Excel file contains 401 rows of engine data, where each row represents a snapshot of engine features.
2. **Data Reshaping:**
   * The loaded data is reshaped to conform to the input shape required by the LSTM layer.
   * For each snapshot, there are 401 time steps and 11 features.
3. **Labeling:**
   * The stable engine data is labeled as class 1, while the unstable engine data is labeled as class 0.
4. **Splitting Data:**
   * The data is split into training and test sets using a ratio of 80% training and 20% testing.

**Training:**

1. **Compilation:**
   * The model is compiled using the Adam optimizer and binary cross-entropy loss function.
   * The accuracy metric is used to monitor model performance during training.
2. **Training:**
   * The model is trained for 10 epochs with a batch size of 32.
   * Training data is fed to the model, and backpropagation updates the model weights to minimize the loss function.

**Evaluation:**

1. **Model Evaluation:**
   * After training, the model is evaluated on the test data to assess its performance.
   * The loss and accuracy metrics are computed to evaluate how well the model generalizes to unseen data.

**Exporting Model:**

1. **Model Saving:**
   * Once trained, the model is saved to a file in HDF5 format using the **save()** method provided by Keras.
   * The saved model can be loaded and reused for inference on new engine data.

**Data Augmentation Process Documentation**

**Overview:**

Data augmentation is a technique used to artificially increase the diversity of the training data by applying various transformations or modifications to the original data. In the context of the provided code, data augmentation is performed on engine data files to create additional augmented files with added noise.

**Augmentation Technique:**

The augmentation technique used in the code involves adding Gaussian noise to the original engine data. Gaussian noise is a random variation with a normal distribution, and adding it to the data introduces small variations that can help the model generalize better during training.

**Implementation:**

1. **Loading Data:**
   * Engine data is loaded from Excel files stored in the "stable" and "unstable" folders using the **load\_data\_from\_folder** function.
2. **Noise Addition:**
   * The **add\_noise** function adds Gaussian noise to the original engine data. The level of noise is controlled by the **noise\_level** parameter, which specifies the standard deviation of the Gaussian distribution. Higher values of **noise\_level** result in more pronounced noise.
3. **Augmentation Loop:**
   * For each file in the "stable" and "unstable" folders, the code reads the data from the file, adds Gaussian noise using the **add\_noise** function, and creates 10 augmented files with noise added.
   * The filename of each augmented file includes an identifier (**augmented\_stable** or **augmented\_unstable**) and a unique index to distinguish between different augmented files.
4. **Output:**
   * The augmented files are saved in the same folders as the original files, with noise-added versions created for each original file. These augmented files can then be used as additional training data to improve the model's performance.