# **README**

## **Project: Lambda Functions Performance Analysis and LSTM Prediction Model**

### Overview:

The project aims to develop and implement an AI-based load balancing framework tailored for serverless architectures in cloud computing. By leveraging machine learning models, specifically LSTM networks, the framework predicts future workloads and optimizes resource allocation to enhance performance, reduce latency, and minimize operational costs.

### Requirements:

* Python 3.x
* Required libraries:
  + pandas
  + numpy
  + scikit-learn
  + tensorflow
  + keras
  + keras-tuner

Install the required libraries:

bash

Copy code

pip install pandas numpy scikit-learn tensorflow keras keras-tuner

### Dataset:

A synthetic dataset lambda\_functions\_sample\_data.csv is generated using random values for performance metrics of AWS Lambda functions, such as:

* FunctionID: Identifier of the Lambda function
* Timestamp: The Unix timestamp of the function invocation
* ResponseTime: The response time of the Lambda function (in seconds)
* CPUUtilization: CPU usage percentage
* MemoryUtilization: Memory usage in MB
* ColdStart: Boolean value indicating whether the function invocation was a cold start
* ColdStartLatency: Latency for cold starts
* ThrottlingRate: Throttling events in the last minute
* ErrorRate: 10% probability of errors

### Steps:

1. Data Generation: A sample dataset is created using random data for the attributes mentioned above and saved as lambda\_functions\_sample\_data.csv.
2. Data Preprocessing:
   * Convert timestamps to cyclical features (Hour of day).
   * One-hot encode the 'FunctionID' column.
   * Normalize features using MinMaxScaler.
   * Save the preprocessed dataset as lambda\_functions\_preprocessed\_data.csv.
3. LSTM Model Training:
   * An LSTM model is trained to predict the Lambda function's ResponseTime based on the other features.
   * Dropout and EarlyStopping techniques are used to prevent overfitting.
   * Save the trained model as lstm\_model.h5.
4. Hyperparameter Tuning:
   * Keras Tuner is used for hyperparameter tuning on the LSTM model, adjusting the number of units, dropout rates, and learning rates.
   * Save the final model after tuning as lstm\_model.keras.
5. Learning Rate Scheduler:
   * A learning rate scheduler is implemented to reduce the learning rate as training progresses.

### Running the Project:

1. Generate Data:
   * Run the script to generate the synthetic dataset.
   * Use the provided data preprocessing steps to prepare the dataset for model training.
2. Train the LSTM Model:
   * Train the LSTM model using the prepared dataset.
   * Apply early stopping to prevent overfitting and restore the best weights.
3. Hyperparameter Tuning:
   * Perform hyperparameter tuning using Keras Tuner to find the optimal model parameters.
4. Evaluate the Model:
   * Evaluate the trained model and make predictions on the test data.

### Files:

* lambda\_functions\_sample\_data.csv: Generated sample data for AWS Lambda function performance.
* lambda\_functions\_preprocessed\_data.csv: Preprocessed and normalized data ready for model training.
* lstm\_model.h5: Saved LSTM model after training.
* lstm\_model.keras: Final LSTM model after hyperparameter tuning and training.